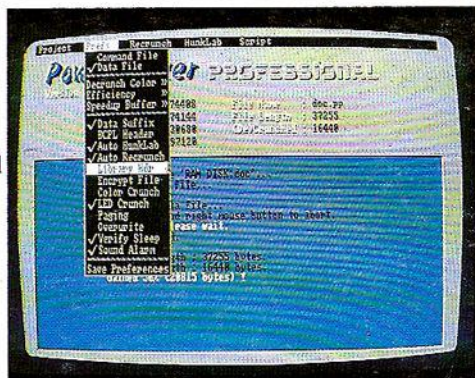
COMPUTER CLUB

88 L. 6.000

La prima rivista per i sistemi Commodore

Anno XI - N. 88 - 25 ottobre 1991 - Sped. Abb. Post. Gr III/70 - CR - Distr.: Parrini

POWER PACKER

Programma risparmia byte

MINIGUIDA

dBase III Plus: impariamolo in un'ora

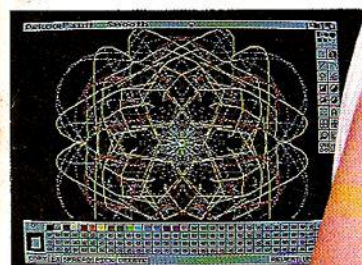
PRIMI PASSI

Programmiamo la stampante

DIDATTICA

Come risolvere ogni equazione

C64

Schermo doppia altezza

DPAINT 4

L'ultima tavolozza

ATTRAZIONE FATALE

Amiga e MS - DOS ora nello stesso disco

Systems

COMPUTER CLUB disco

1

La prima rivista per i sistemi Commodore

CORRI A COMPRARLO É IN EDICOLA!

TUTTO AMIGA

Lire 10.000

IN REGALO:

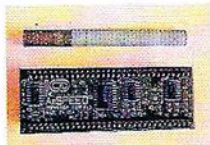
Un fantastico programma AMIGA per gestire i dischi MS - DOS

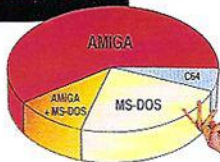
Programmi...
Icone...
"Virus"...
Antivirus...
Game...
Matematica...
e tanto altro ancora!

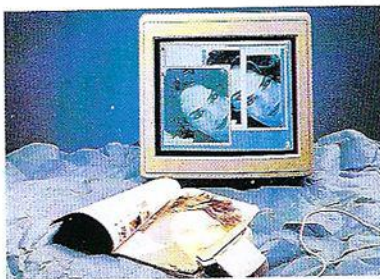
COMPUTER CLUB # 87 L. 6.000

La prima rivista per i sistemi Commodore

Anno XI - N. 87 - 25 agosto-25 settembre 1991 - Sped. Abb. Post. Gr III/70 - CR - Distr.: Parrini

A 500
Minischeda acceleratrice

INCHIESTA
Ecco, come siamo

AMIGA
C64
AMIGA + MS-DOS
MS-DOS

AMIGA
- Solidi in rotazione
- Grafica in Kick-Pascal
- Assembly: gli indirizzamenti

STAMPANTI
Dalla carta al video e ritorno

Briscola con C64, Amiga ed Ms-Dos

La vostra posta • I libri di Ventura • Emulatore di virus • La sfida di Pitagora • Contatempo in Turbo Pascal • Assembly 80X86 • Enciclopedia Ms-Dos • Amiga C • Inchiesta sulle stampanti • La tua foto nel computer

Systems

Sommario 88

Spazio C/64

Amiga

Mondo Ms - Dos

Amiga + MS - DOS

COMPUTER CLUB

Direttore: Alessandro de Simone

Coordinatore: Marco Miotti

Redazione/collaboratori:
Davide Ardizzone - Claudio Baiocchi
Luigi Callegari - Umberto Colapicchioni
Donato De Luca - Carlo d'Ippolito
Valerio Ferri - Michele Maggi
Giancarlo Mariani - Ascanio Orlandini
Domenico Pavone - Armando Sforzi
Dario Pistella - Fabio Sorgato
Valentino Spataro - Franco Rodella
Stefano Somonelli - Luca Viola

Direzione:
Via Mosè, 22 cap. 20090 OPERA (Mi)
Telefono 02/ 57.60.63.10
Fax 02/ 57.60.30.39
BBS 02/ 57.60.52.11

Pubblicità:
Leandro Nencioni (dir. vendite)
Via Mosè, 22-20090 Opera (Mi)
tel. 02/ 57.60.63.10

Emilia Romagna:
Spazio E
P.zza Roosvelt, 4 cap. 40123 Bologna
Tel. 051/ 23.69.79

Toscana, Marche, Umbria
Mercurio s.r.l. Via Rodari, 9
S. G.nni Valdarno (Ar)
Tel. 055/ 94.74.44

Lazio, Campania
Spazio Nuovo
Via P. Foscari, 70 - cap. 00139 Roma
tel. 06/ 81.09.679

Abbonamenti: Liliana Spina
Arretrati e s/w: Lucia Dominoni

Tariffe: Prezzo per copia L. 6000
Abbonamento annuo (11 fascicoli) L. 60000
Estero: L. 100000 - Indirizzare versamenti a:
Systems Editoriale Srl
c/c 37952207 oppure inviare come assegno bancario non trasferibile e barrato due volte a:
Systems Editoriale Srl (servizio arretrati)
Via Mosé, 22
cap. 20090 OPERA (Mi)

Composizione e fotolito:
Systems Editoriale

Stampa:
La Litografica Srl Cuggiono (Mi)

Registrazione: Tribunale di Milano
n.370 del 2/10/82

Direttore Responsabile:
Michele Di Pisa

Spedizioni in abbonamento postale gruppo III.
Pubblicità inferiore al 70%

Distrubutore:
Parrini - Milano

Periodici Systems:
Amiga Club (disco ed. Germania) - Banca Oggi - Computer (quotidiano) - Computer Club - 64 Club (disco) - Computer Club (disco ed. Germania) - Hospital Management - Nursing '90 - PC Club (disco ed. Germania) - Personal Computer - Jonathan - VR - Videoteca

Editoriale

Un supporto indispensabile

L'inchiesta condotta sui numeri scorsi ci induce a tentare un'altra via per la diffusione della cultura informatica.

Ci riferiamo, in particolare, alla necessità di eliminare, una volta per tutte, l'inconcepibile divario che separa due categorie di utenti, inconsapevolmente legate dallo stesso motivo conduttore: la passione per l'informatica ed il progresso in senso lato.

Ne è la riprova l'ormai quasi totale assenza di lettere inviate da 64isti (a proposito, sul prossimo numero pubblicheremo l'ultima sequenza di polemiche sul *"computer più bello del mondo"*) che lamentavano la ridotta presenza di articoli che li riguardavano direttamente.

Al momento in cui scrivo leggo infatti che sul listino di Flopperia, un notissimo Computer Shop milanese, un **C/64** dotato di drive originale 1541 viene posto in vendita a **540 mila lire** e con drive compatibile a **440 mila**; un computer MS - DOS (processore V20, tastiera, drive 5.25, controller, scheda video + printer, 521 Kram, cabinet) viene invece offerto a sole **399 mila**: chi avrà il coraggio, oggi come oggi, di spendere una cifra maggiore per procurarsi un vecchio computer ad 8 bit?

Diverse risposte all'inchiesta sulle stampanti, comunque, ci fa intuire che, almeno su alcuni argomenti, le idee non sono ancora molto chiare: come si può, ad esempio, pretendere di "fare" del DTP affermando, nel contempo, di non voler cambiare la propria stampante a 9 aghi?

Ecco, quindi, che interviene un'altra pubblicazione, **Computer Club Disco** appunto, che offre l'opportunità, anzitutto, di affrettare la demolizione del muro che separa Amiga da MS - DOS; inoltre dovrebbe consentire la divulgazione di tecniche di programmazione efficacemente utili per entrambi i sistemi.

Infine, ambizione non ultima, si cercherà di realizzare ciò che, in Italia, nessuno ha mai proposto in modo sistematico: la formazione di una equipe, quanto più vasta possibile, di autori *italiani* di pubblico dominio in grado di realizzare sia programmi di utilità generale sia archivi di interesse universale, che prescindano dalla macchina su cui vengono adoperati.

In parole più semplici, ciò significa che il nostro sogno è quello di offrire, su disco, procedure di trattamento di testi, di immagini, di suoni; archivi statistici, geografici e quadri sinottici di storia, filosofia, arte, letteratura; intere opere letterarie, in formato *compresso*, a disposizione degli utenti della scuola. Il tutto, ovviamente, in un formato che consenta, a chiunque lo voglia, di accedere rapidamente all'informazione cercata, usando, magari, un'utility che ne faciliti la ricerca stessa.

Inutile dire che, in questo sogno ambizioso, il protagonista assoluto è lo stesso utente di **Computer Club Disco**, motivato allo sviluppo di programmi ed archivi dalla passione per l'informatica che ci accomuna tutti.

Un'utopia destinata, come tale, a restare sulla carta?

Il primo passo per una conferma, positiva o negativa che sia, l'abbiamo comunque fatto.

A voi il compito di stendere il verdetto.

Alessandro de Simone

LA VOSTRA POSTA

(a cura di A. de Simone)

File acefalo

Con il mio C/64 avevo registrato un file di testo ASCII su un nastro cassetta; in seguito, per errore, registrando un programma troppo lungo ho cancellato l'intestazione del file, ma questo sembra essere integro, almeno ascoltando il nastro con un normale registratore. Posso recuperare, anche se in parte, il contenuto del file suddetto?
(Marco Bianchi)

Purtroppo no. A differenza dei file registrati su disco, quelli memorizzati su nastro hanno un assoluto bisogno dell'intestazione che consente, al sistema operativo, di sincronizzarsi correttamente con la velocità di scorrimento del nastro.

Operando con i **dischetti**, invece, se vengono cancellate parti anche consistenti di file, ciò che avanza può essere recuperato, pur se a prezzo di notevol(issim)i sacrifici e sperando di riuscire nell'intento.

E' sufficiente(!), infatti, esaminare traccia per traccia e settore per settore, l'intera superficie del disco servendosi di un comune editor di dischi (ad esempio: **Disk Master** per Amiga oppure **Pc Tools** per MS - DOS). Se il dischetto contiene pochi file e non sono state compiute, su di esso, eccessive sovrascritture, il problema è risolvibile a livello umano.

Se, invece, devi rintracciare spezzoni di file all'interno di un **hard disk** da 100 megabyte, beh, meglio rinunciare non appena incominci a vedere vermi rossi che strisciano lungo le pareti.

Non ti azzardare, comunque, a recuperare file di programma parzialmente sovrascritti:

diventeresti pazzo dopo pochissimi minuti!

Parentesi in C

Lavorando con l'editor (al "di fuori", invece, non vi sono problemi di sorta) del linguaggio Turbo C con un computer MS - DOS mi capita spesso di utilizzare parentesi graffe che, non essendo presenti sulla tastiera, sono costretto a far apparire premendo il tasto ALT insieme alle cifre 1 - 2 - 3 oppure 1 - 2 - 5. Non c'è un sistema più rapido, magari programmando alcuni tasti?
(Nunzio Santini - Comiso)

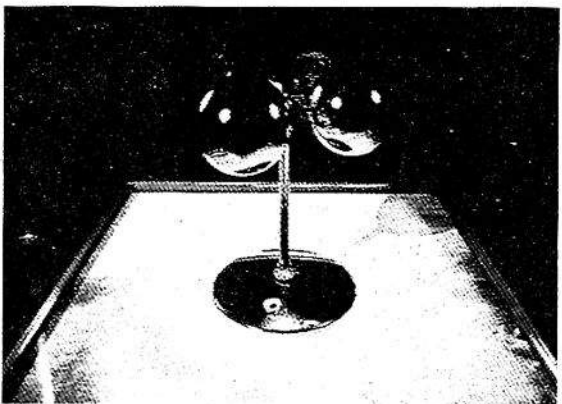
Vi sono diversi sistemi, ma tutti molto più complicati della pressione dei quattro tasti citati.

A dire il vero, non ho mai tentato di effettuare esperimenti del genere perché sono talmente abituato all'uso di ALT che continuerei ad usare questo sistema anche se avessi a disposizione tasti pre-programmati.

Comunque, se qualcuno vuol dare un suggerimento al nostro affezionato lettore, lo pubblicheremo con piacere.

Il più veloce

Alcuni appassionati affermano che il computer più veloce è l'Amiga; altri, invece, il Macintosh. A chi credere? (e non rispondetemi: a tutti e due).
(F. Geromel - Roncade)

A tutti e tre (hai dimenticato MS - DOS); la risposta, apparentemente ironica, non può non tener conto di varie

considerazioni, vitali per una risposta corretta.

Il fatto che Amiga e Macintosh montino processori della stessa famiglia (**68000**) non significa che, almeno in teoria, le prestazioni siano simili.

Anni fa, tanto per fare un esempio, Apple e Commodore utilizzavano lo stesso processore (**6502**) sui computer che producevano, ma le differenti caratteristiche dei vari modelli erano tali da scavare solchi profondi tra i computerologhi dell'epoca.

Anche oggi, ahinoi, bande armate di affezionati fanno discorsi di sapore manicheistico pretendendo di asserire che il computer per eccellenza è un certo modello e non altri.

Non ci si rende conto che, in certi casi, un modello è superiore ad un altro; in altri casi, invece, è meglio rinunciare al suo uso.

L'aereo è molto veloce, ma spesso è più comoda l'automobile; per problemi di parcheggio in centro il mezzo pubblico è però insostituibile; se il tempo è bello, poi, la bicicletta è anche divertente; che dire, poi, della barca, del treno e... dei piedi?

Insomma, a seconda dei casi un sistema è migliore di un altro.

Certo, non tutti possono permettersi il lusso di avere a disposizione tre computer; ciò non toglie valore, comunque, al ragionamento di fondo.

Per quanto riguarda la gestione della grafica, argomento principale della lettera inviata, Amiga dispone di uno specifico chip realizzato proprio tenendo conto di un utilizzo intensivo nel campo ludico e, in particolare, dell'animazione.

Se, quindi, alla manifestazione Bit Movie (di cui Callegari ha parlato nel N. 86 di C.C.) 29 delle 30 animazioni erano realizzate con Amiga, beh, un motivo ci sarà senz'altro.

Riviste gratis

Ho ricevuto, gratuitamente, un pacco di riviste della vostra casa editrice. A che devo il piacere di un simile regalo?

(Francesco Varone - Bellona)

La nostra modesta(!) generosità deve essere considerata un incoraggiamento verso quei lettori che dimostrano di seguirci con una certa attenzione o che partecipano attivamente, comunque, all'attività informatica.

E' probabile, nel caso specifico, che il nominativo sia stato estratto a sorte tra le risposte alle inchieste che pubblichiamo mensilmente; oppure che la risposta ad una sfida, magari non pubblicata, è stata tuttavia considerata meritevole di un certo riconoscimento; oppure come premio di consolazione per l'invio di un dischetto contenente software (purtroppo) inadeguato alla pubblicazione.

La Systems Editoriale, come qualsiasi altra casa editrice, non ha alcun dovere di compensare gli sforzi dei suoi lettori; tuttavia ci sia consentito esprimere, quando capita, un parziale ringraziamento alla loro fedeltà e bravura manifestata in varie occasioni.

Differenze 80X86

Che differenza c'è tra i processori 80286, 80386DX, 80386SX, 80486?

(da alcune lettere)

Il processore 80286 è l'evoluzione della precedente serie 8088 e 8086 che, all'avanguardia all'epoca della loro progettazione, caddero in disgrazia non appena i programmi iniziarono ad essere avidi di memoria RAM.

I vecchi processori, infatti, non riuscivano a *vedere* oltre i 640 Kram, limitazione in parte superata attraverso una macchinosa gestione della memoria da parte del più nuovo 80286 che, per l'occasione, vantava una maggiore velocità operativa ed un incremento delle istruzioni a disposizione.

La stessa difficoltà di gestione della memoria, oltre alla continua necessità di avere a disposizione masse sempre più estese di memoria RAM, indusse la Intel a progettare il micro **80386**, che risolveva tutti i problemi.

Ci si accorse che un suo sotto-modello (l'80386**sx**, appunto) poteva tuttavia essere prodotto a **prezzi più bassi** e proposto per realizzare computer in cui l'esigenza di una gestione facilitata della memoria era prevalente rispetto ad altre caratteristiche.

Un computer basato su un 80**386sx**, paradossalmente, risulta infatti spesso meno efficiente di uno basato su un più modesto 80**286**, che di solito è più veloce nel compiere una grande quantità di operazioni.

Il micro 80386**dx**, invece, è tutt'altra cosa dal momento che dispone di un bus di indirizzi **doppio** (da cui la sigla **dx** rispetto all'altra, sx, che sta per **singolo**) e compie le operazioni di accesso alla memoria in un tempo sensibilmente più breve.

Il processore 80**486**, infine, dispone al suo interno anche di un **processore matematico** e di una cache memory che, proprio grazie alla loro ubica-

Emulatore C/64 in ambiente MS - DOS

Su una banca dati abbiamo individuato, tempo fa, un programma divertentissimo che emula il C/64 su un computer MS - DOS.

Si tratta di una procedura (scritta in tedesco!) che, una volta attivata, **trasforma un computer MS - DOS in un C/64**. Ciò significa che è non solo in grado di interpretare i vari comandi del Basic V.2, ma addirittura permette di operare in linguaggio macchina 6502 utilizzando un **Monitor**, compreso nel "pacchetto".

Per verificarlo, abbiamo scritto vari programmi Assembler che interagivano con la mappa dello schermo ed operato, da Basic, mediante le più note **Poke** dell'obsoleto computer: tutto funziona a meraviglia, compresa l'emulazione del ben noto **Run / Stop e Restore**.

Il problema maggiore, però, è rappresentato dalla possibilità di memorizzare e caricare programmi.

Della procedura fa parte un'area di circa 160 Kbyte che emula, in tutto e per tutto, la superficie magnetica di un dischetto inserito nel drive 1541. Se, quindi, si registra, si verifica, si formatta il "disco", l'area si comporta esattamente come se fosse un 1541.

Purtroppo non è possibile leggere veri dischetti di formato 1541; sembra, infatti, che l'unico modo di caricare programmi sia quello di scrivere un programma di telecomunicazione e mettersi in contatto (via modem) con una banca dati o (mediante interfaccia Rs-232) con un vero C/64 opportunamente predisposto per la trasmissione di programmi e dati.

Una curiosità, insomma, e nient'altro.

Una sfida... parasportiva!

Anche se, in altra parte di questo stesso fascicolo, è lanciata la tradizionale sfida, in queste pagine tenteremo di invogliare i lettori più bravi a cimentarsi in procedure che, sicuramente troppo complesse per esser pubblicate sulle pagine di questa rivista, possono trovare posto sulla neonata "**Computer Club Disco**", in cui lo spazio non costituisce certo un problema.

In questo primo appuntamento abbiamo rubato un po' di spazio alla rubrica della posta, anche perché la procedura che stiamo per proporre non richiede molte parole.

Per fortuna, infatti, la sua descrizione è piuttosto semplice, anche se la sua realizzazione non è allo stesso livello di semplicità. Ma bando alla ciance.

Avete presente la notissima rubrica televisiva dal titolo "**Novantesimo minuto**"?

Bene, fate caso alle due animazioni che vengono spesso replicate durante la trasmissione: nella prima, lo schermo che raffigura un pallone di calcio su fondo verde, viene diviso in due parti, orizzontali, che si allontanano verticalmente per lasciare lo spazio ad una schermata sottostante.

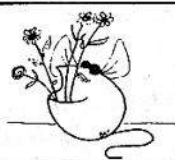
Si tratta, quindi, di scrivere una procedura che, utilizzando due schermate grafiche già presenti in memoria o su disco (di qualunque tipo esse siano) simuli l'animazione prima descritta.

Visto che ci siete (ma ci rivolgiamo solo ai più bravi) pensate ad una procedura che divida lo schermo in due parti orizzontali, oltre che verticali, e/o che consenta di dividere in due anche la seconda schermata (per far riapparire nuovamente la prima).

La seconda animazione è molto più difficile: vi chiediamo di simulare il numero **90** (sigla, appunto, di *Novantesimo minuto*) che ruota attorno ad un sfera (di solito la si può notare nello spigolo in alto a destra durante la trasmissione domenicale). Naturalmente sarebbe opportuno scrivere una procedura che riesca a riprodurre fedelmente i movimenti prospettici non solo del numero 90, ma di una qualunque **brush** precedentemente registrata, ad esempio, con un qualsiasi programma grafico.

Stavolta vi chiediamo uno sforzo davvero non comune; ma l'elevato livello medio dimostrato, in più occasioni, dai nostri lettori ci induce a sospettare che qualcuno sarà certamente in grado di rispondere alla sfida.

Cercate di non smentirci, altrimenti che figura facciamo?...

zione all'interno di un **unico circuito integrato**, consente di incrementare ulteriormente la velocità operativa. E' noto, infatti, che la lunghezza delle piste di rame influisce negativamente sulle prestazioni di un sistema: ecco il motivo per cui si tende a miniaturizzare i circuiti o, meglio, incorporare in un unico *case* le funzioni di una pluralità di circuiti integrati. In conclusione, quindi, il migliore 80X86 è inevitabilmente l'80486, seguito dall'80386dx e (a pari merito? Fate un po' voi...) dall'80286 e 80386sx. Si tenga conto, comunque, che si sta verificando un sensibile abbassamento dei prezzi dei processori.

WordPerfect per tutti

WordPerfect non è solo un word processor, ma molto di più. Non ci soffermeremo certo sulla possibilità che offre per quanto riguarda la stesura di documenti che, come Word della Microsoft ed altri pacchetti altrettanto blasonati, esce dagli angusti limiti di un w/p per entrare, di diritto, nel mondo **Desk Top Publishing**.

La caratteristica peculiare di WordPerfect è relativa alla possibilità di operare allo stesso modo, o quasi, su tutte le piattaforme oggi disponibili nel campo dell'informatica. Da **Amiga** a **Ms-Dos**; da **Macintosh** a **Next**; dai piccolissimi **Atari** al gigante **IBM 370** ed ai sistemi **Vax**; in **rete** o **stand alone**; per ogni "aggeggio" dotato di tastiera, insomma, è presente una proposta WordPerfect che sembra dirigersi verso uno standard di primaria importanza nel campo della editoria in senso generale. I prezzi di listino, come purtroppo è abitudine nel campo del software professionale, non sono proprio a livello hobbistico. Fortunatamente, tuttavia, viene offerta una speciale promozione che vede, come diretti interessati, coloro che operano nel campo educazionale, tra cui anche gli studenti stessi.

E' sufficiente inviare una certificazione da cui risulta lo "stato" di studente (o professore) per ottenere una sensibile riduzione del prezzo di vendita: la versione Amiga, in questo caso, passa da 400 mila a **160 mila**; quella Ms-Dos da oltre un milione a **315 mila**; quella per Macintosh da oltre 900 mila a meno di 400 mila e così via.

Per ulteriori informazioni:

WordPerfect Italia
Corso Sempione 2
20154 Milano
Tel. 02 / 33.10.62.00
Fax. 02 / 33.10.61.90

✔ Amiga	❑ Principianti	✔ Esperti	❑ Tutti	✔ HELP
✔ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
✔ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
❑ Assembly				

Una logica conseguenza della sfida lanciata qualche mese fa.

Tutte le sillabe italiane

< Una sfida al mese > < Esaminare C. C. n. 83 e 77 >

Sul numero di *aprile* invitammo i lettori a scrivere un algoritmo di trattamento di testi in cui inserire una routine di divisione in sillabe.

Alla sfida risposero vari lettori (tra cui **Francesco Varone**, di cui pubblichiamo, a parte, il lavoro), tutti abbastanza bravi da risolvere in modo soddisfacente il problema.

Altri lettori altrettanto capaci decisero di partecipare alla sfida, ma presentarono i propri lavori con notevole ritardo e conseguente esclusione dal "concorso".

Diamo ora una nuova opportunità a tutti i lettori, purché veramente capaci, dal momento che la sfida di questo mese richiede uno sforzo maggiore, che non consente alcuna improvvisazione.

La sfida

Il programma richiesto ha lo scopo di **individuare, e memorizzare, tutte le sillabe disponibili nella lingua italiana.**

Si tratta, in pratica, di realizzare una specie di *"alfabeto sillabico"* (scusate l'orrendo neologismo) contenente tutte le sillabe in grado di comporre una qualsiasi parola del nostro idioma. Una routine di divisione in sillabe (come quella pubblicata sullo stesso **n. 83**) dovrà, per forza di cose, essere la protagonista indiscussa della procedura che, semplificando il concetto, deve funzionare in questo modo:

Dopo l'attivazione del programma deve comparire una richiesta simile alla seguente: *Quale file devo elaborare?*.

Tale domanda si riferirà ad un file ASCII, memorizzato in precedenza su disco, contenente un qualsiasi documento scritto in italiano con un comune text editor (e **non** con un word processor!), cioè privo di caratteri speciali.

A questo punto inizia l'elaborazione vera e propria, vale a dire:

➪ Lettura, byte per byte, del file ASCII con esclusione di caratteri di punteggia-

tura (virgole, punti, punti e virgole, esclamativi, interrogativi, apostrofi, eccetera), esclusione di caratteri numerici e, comunque, non alfabetici e conversione dei caratteri maiuscoli in minuscoli, o viceversa, allo scopo di evitare equivoci o doppioni.

➪ Individuazione di ogni singola parola, grazie al carattere separatore di spazio o return, ed elaborazione della routine di divisione in sillabe con relativa memorizzazione, in un vettore, di tutte le sillabe individuate.

➪ Confronto di ciascuna sillaba individuata con quelle già memorizzate in precedenza in un apposito vettore (se, ovviamente, il programma ha girato più di una volta). Nel caso venga individuata una sillaba sconosciuta, inserirla, in ordine alfabetico, nel vettore-archivio per arricchire l'alfabeto sillabico.

Supponiamo che il file ASCII di testo contenga la seguente frase:

Giovedì, 15 ottobre, era una giornata molto luminosa; ho telefonato a Gianni Misato.

Dopo una prima filtratura, la frase diventa...

giovedì ottobre era una giornata molto luminosa ho telefonato a gianni misato

Alla fine dell'elaborazione, il vettore relativo all'alfabeto sillabico finora costruito conterrà le sillabe inserite nel riquadro riportato in queste pagine.

E' ovvio che il numero di sillabe, all'inizio delle operazioni, aumenterà molto rapidamente ma, con il passare del tempo (cioè dando "in pasto" al programma numerosi file ASCII di testo) tenderà a stabilizzarsi, fino a che gli aggiornamenti dell'alfabeto sillabico diventeranno un episodio piuttosto raro.

a	ho	ot
bre	le	ra
dì	lu	ta
e	mi	te
fo	mol	to
gian	na	sa
gio	ni	u
gior	no	ve

Sillabe contenute nella frase di esempio.

A che serve?

Mi sembra di sentire la classica domanda che un programmatore si sente in dovere di porre: a che serve un programma del genere?

Beh, anzitutto per mettere alla prova la vostra bravura. Se, infatti, l'algoritmo di "filtro" dei caratteri non presenta particolari difficoltà, non pensate che la routine di gestione dell'alfabeto sillabico sia altrettanto semplice.

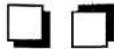
Il nostro suggerimento è quello di far caricare, all'inizio della procedura, l'alfabeto sillabico (eventualmente realizzato fino a quel momento) in RAM e di applicarvi una routine di inserimento, e successivo ordinamento alfabetico, non appena viene individuata una sillaba nuova.

Routine di ordinamenti alfabetici sono pubblicate su qualsiasi manuale di programmazione, anche scolastici. Computer Club ne ha comunque divulgate parecchie, tra cui quelle in Gw-Basic sul **N. 77** (articolo *"C'era una volta il Caos; poi fu Sort"*).

Un utilizzo pratico dell'alfabeto sillabico potrebbe riguardare, inoltre, eventuali routine di **compressione di testi** ASCII.

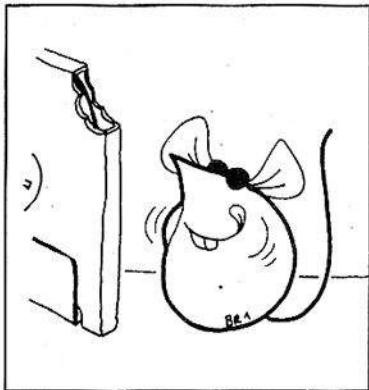
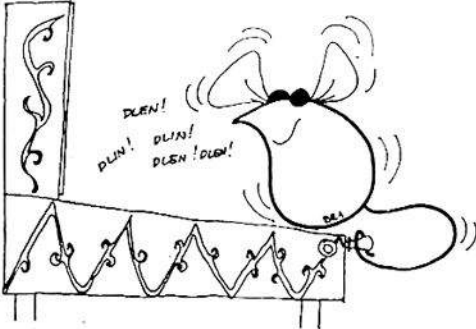
Supponendo, infatti, che tutte le sillabe italiane siano poche centinaia, si potrebbe elaborare un particolare sistema di codifica in cui, ad ogni numero, corrisponda una certa sillaba.

Un'altra applicazione, pur se specifica, potrebbe esser quella di facilitare la stesura di... **parole crociate** con schemi sillabici!

Fino a che, tuttavia, non vi saranno dati sufficienti sul numero effettivo di sillabe disponibili nel nostro idioma, immaginare una pratica applicazione è prematuro.

Per i più bravi

I lettori più capaci potranno cimentarsi nel realizzare, anziché un vettore, una **matrice bidimensionale** in cui affiancare, a ciascuna sillaba, un vocabolo in cui la sillaba stessa è utilizzata.

Ad esempio, le prime sillabe del riquadro potrebbero essere...

a a
bre ottobre
dì giovedì

In questo modo sarebbe più facile individuare sillabe appartenenti a vocaboli digitati in modo non corretto (gio**veì**, lumi**noxa** e così via) oppure stranieri (es. good morning).

Anche stavolta il premio in palio è da favola: **all'autore del miglior lavoro pubblicato verrà consegnato un documento di viaggio, valido per una settimana, in una prestigiosa metropoli europea.**

Per favorire il vincitore, precisiamo che il biglietto settimanale dell'azienda tranviaria milanese ATM, valido per circolare su tutte linee, potrà essere inviato, a richiesta del vincitore, presso il suo domicilio...

Come partecipare

Nelle sfide mensili lanciate da Computer Club viene sempre data precedenza ai programmi, scritti in qualsiasi linguaggio, in grado di girare su Amiga oppure computer MS-DOS compatibili. I listati devono essere strutturati in modo chiaro, allo scopo di favorire eventuali modifiche da parte di lettori che intendano sofisticare la procedura. Listati ed articolo esplicativo devono tassativamente essere inviati su disco (formato 3.5 o 5.25) oppure, via modem, alla nostra BBS (tel. 02/57.60.52.11).

In ogni caso, onde evitare invio di materiale inadeguato alla pubblicazione, si consiglia di telefonare in redazione (Tel. **02/57.60.63.10**) per concordare l'inoltro del lavoro svolto.

- ❑ Amiga
- ❑ Ms - Dos
- ✔ C/64
- ❑ Hardware
- ❑ Software
- ✔ Grafica
- ✔ Programmazione
  - ❑ Dos
  - ❑ Pascal
  - ❑ C
  - ❑ Basic
  - ❑ Assembly

❑ Principianti ✔ Esperti ❑ Tutti ✔ HELP

Una procedure in lm che sarà apprezzata dagli esperti del glorioso C/64

Uno schermo rimbalzante

< di Filippo Bruno >

< Uno degli ultimi articoli sul C/64! >

Del C/64, tra breve, non ci occuperemo più, per una serie di motivi ormai ben noti.

E' per noi abbastanza complesso trovare ancora qualche argomento nuovo che sia in grado di attirare l'interesse gli "irriducibili".

Lo scopo dell'articolo di questo mese è di arricchire la propria biblioteca di routine in **linguaggio macchina** con un programma che conferisce un tono professionale alle vostre creazioni.

La routine effettua la sostituzione tra due schermate in bassa risoluzione, cioè contenenti caratteri alfanumerici e simboli, con un **effetto di rimbalzo smorzato**, simile a quello di una pallina di gomma lanciata sul pavimento.

La schermata originale, che chiameremo **Screen 1** per comodità di riferimento, al momento del Run scorrerà verso il basso mentre lo spazio vuoto lasciato in alto verrà gradualmente occupato dalla seconda schermata (che chiameremo **Screen 2**).

Screen 1 non cederà subito il posto allo Screen 2 ma impiegherà 5 cicli per scomparire definitivamente dallo schermo visibile, intendendo per **ciclo** un rimbalzo che avverrà ogni qualvolta il fondo dello Screen 2 raggiungerà il fondo visibile del monitor.

La tecnica di rimbalzo è relativamente semplice e sfrutta intensamente il **Raster Register** per evitare indesiderati sfarfallii ed asincronismi che andrebbero a discapito della fluidità dello scrolling. C'è anche la possibilità di ottenere un **effetto sonoro** per l'intera durata della sovrapposizione tra le due schermate, che comunque può essere eliminato osservando attentamente le istruzioni poste all'interno delle Rem nel programma Basic caricatore.

Come gira

Il listato Basic si incarica di allocare in memoria i dati per la routine; rimandiamo oltre la spiegazione dettagliata sul funzionamento, in modo da accontentare coloro che fossero solo interessati alla fruizione e non anche alla comprensione del programma.

Il programma **Basic** inizia il suo lavoro allocando i **codici** in lm, e la **tabella** necessaria ad essi, nella parte della memoria al riparo da possibili interferenze con l'interprete Basic o con routine poste in ROM (zona da $C000 a $D000, ossia da **49152** a **53248**); verifica che i dati letti, contenuti nei Data, siano corretti, segnalando eventuali errori di digitazione.

In seguito chiede di riempire lo schermo a piacimento, schermo che formerà lo Screen 2; occorre tenere presente varie limitazioni, alcune eliminabili, altre no:

➪ la limitazione non eliminabile è costituita dal fatto che la prima riga dello Screen 2 non verrà mai visualizzata perché, per ottenere uno scrolling fluido, è necessario impostare la visualizzazione a **24** righe e non **25**, come di default: in pratica la prima riga viene cancellata in quanto il bordo superiore la ingloba interamente.

➪ la limitazione eliminabile è invece costituita dal fatto che, sia perché la routine di riempimento del video è "manuale" (cioè gestita da un Input), sia perché ci troviamo in modo diretto, non sarà possibile scrivere sull'ultimo carattere del video (riga 25, colonna 39) in quanto lo schermo scorrerà inesorabilmente una o due righe in alto, cancellando parte di ciò che avevamo scritto nelle prime righe e comunque sfasando il nostro lavoro.

Inoltre avremo seri problemi nell'inserimento di caratteri proibiti come la virgola, il punto e virgola, le virgolette e simili: al momento della pressione del tasto Return, infatti, potremmo avere la spiacevole sorpresa di veder comparire sullo schermo il messaggio *Redo From Start?* che costringerebbe a ripetere le opera-

```
;********************************
;*     schermo rimbalzante      *
;* written on  september 1991 by  *
;*         f.brl jr soft          *
;*  alias: filippo bruno  *roma*  *
;********************************
;
;--------------------------------
        raster = $d012
        hirast = $d011
;--------------------------------
        *=$c000             ;start=49152
move    tay                 ;'salva' a in y
        ldx #$00            ;prepara la locazione $fc
        stx $fc             ;per la moltiplicazione
        and #$07            ;inserisce il valore di scroll
        ora #$10            ;nell'apposito registro
        sta hirast          ;e setta bit 3 per evitare blank
        tya                 ;recupera a
        and #$f8            ;considera i bit 7->3
        sta $fb             ;moltiplica questo valore
        asl a               ;per 4 e lo forza
        rol $fc             ;nei puntatori $fb/$fc
        asl a               ;nel formato basso/alto
        rol $fc             ;
        clc                 ;vi riaggiunge la quantita'
        adc #$fb            ;originaria->viene moltiplicato
        sta $fb             ;in totale per 5
        lda $fc             ;aggiunge in $fc
        adc #$00            ;l'eventuale carry
        sta $fc             ;
        sec                 ;setta il carry per sottrazione
        lda #$66            ;prepara i puntatori $fb/$fc
        sbc $fb             ;riempiendoli con un
        sta $fb             ;valore uguale a
        lda #$c6            ;$c666-#40*n
        sbc $fc             ;con 6<n<31
        sta $fc             ;
        lda #$00            ;prepara i puntatori $fd/$fe
        sta $fd             ;facendoli puntare all'area
        lda #$04            ;video ($0400)
        sta $fe             ;
        ldy #$00            ;riempie lo schermo
trasf   lda ($fb),y         ;con parte di memoria (screen 1)
        sta ($fd),y         ;salvata all'inizio (jsr copy)
        iny                 ;e parte della presente (screen 2)
        bne trasf           ;ripete un ciclo di 255 caratteri
        inc $fc             ;finche' $fe non e' 8
        inc $fe             ;cioe' abbiamo riempito la
        lda $fe             ;zona fino a $0800 (escluso)
        cmp #$08            ;da dove inizia l'area basic
        bne trasf           ;
        rts                 ;torna
copy    ldx #$00            ;copia tutto cio' che e'
loop    lda $0400,x         ;presente sullo schermo (screen 1)
        sta $c576,x         ;nella parte di memoria
        lda $0500,x         ;subito successiva a quella
```

zioni. Questi problemi si possono aggirare evitando di utilizzare la tecnica usata nel programma Basic di queste pagine, puramente dimostrativo, e notando che i dati - video relativi sia allo Screen 1 che allo Screen 2 sono necessariamente presenti in memoria a partire, rispettivamente, da $C18E (**49550**) e $C576 (**50550**).

E' quindi possibile inserire manualmente, o addirittura tramite un opportuno *Load*, i dati nelle aree richieste.

Occorre prestare attenzione al fatto che la routine, così come è formulata, copia automaticamente il contenuto dello schermo al momento della SYS, o del lancio tramite JMP all'interno di un altro programma in lm.

Per impostare entrambi gli Screen, quindi, occorre cancellare il salto a Copy nel disassemblato (magari sostituendo i 3 byte necessari con 3 byte contenenti il numero 234 (**$EA**) che corrisponde all'istruzione NOP e provvedendo al riempimento dell'area dello Screen 1 in precedenza.

Dopo il rimbalzo, il programma chiederà se preferite cambiare ancora lo Screen 2 o se invece volete **salvare su disco** la routine in lm, compreso lo Screen 2; in quest'ultimo caso, dopo l'operazione di salvataggio, su disco sarà presente un file che, caricato con la sintassi **Load "nome File", 8, 1**, verrà automaticamente allocato a partire da $C000 (49152).

Come funziona

Passiamo ora alla spiegazione più dettagliata della routine.

Essa, benché allocata a partire da $C000, inizia da $C069 (**49257**), in quanto precedentemente sono allocate alcune subroutine. Per prima cosa, come già spiegato, copia il contenuto dello schermo nell'area dello Screen 1. In seguito legge i **255** dati della tabella saltando alla subroutine finalizzata allo spostamento dei caratteri (subroutine **Move**) dopo ogni dato letto.

E' importante notare che, prima di ogni salto a Move, la routine attende che il pennello elettronico abbia terminato il lavoro di scandaglio e si sia riposizionato all'inizio del video sulla linea 0 (posta sotto il bordo).

```
        sta $c676,x       ;che contiene i valori
        lda $0600,x       ;della sec.a schermata (screen 2)
        sta $c776,x       ;
        lda $06e8,x       ;
        sta $c85e,x       ;
        inx               ;
        bne loop          ;
        rts               ;torna
start   jsr copy          ;salva screen 1
        sei               ;disabilita irqs
        lda #$00          ;utilizza la locazione $02 come
        sta $02           ;registro x stabile
again   ldx $02           ;inizia il ciclo
        lda $c08e,x       ;legge il valore n.x da tabella
        ldx #$00          ;aspetta che il raster
cntrl   cpx raster        ;sia giunto alla
        bne cntrl         ;riga 0 (sotto il bordo) e che
        ldy hirast        ;il bit 7 di hirast sia spento
        bpl cntrl         ;
        jsr move          ;esegue lo spostamento
        inc $02           ;incrementa contatore stabile $02
        lda $02           ;finche' e' <> da 0
        bne again         ;continua
        cli               ;ripristina irqs
        rts               ;torna
        nop               ;
;---------------------------------
        .end              ;fine del prg
```

Ciò per evitare asincronismi dovuti ad erronee anticipazioni. La subroutine Move non fa altro che elaborare il dato letto dalla tabella ed interpretarlo correttamente.

Analizziamo ora il modo in cui il C/64 gestisce gli **scrolling.** Una particolare locazione di memoria ($D011 - **53265**) mette a disposizione 4 bit per tale scopo: i bit 0, 1 e 2 sovraintendono lo scrolling verticale dell'intero schermo, mentre il bit 3 imposta il modo 24 righe. Ma agendo solo sui 3 bit dello scrolling non si può ottenere lo scrolling del video di 25 * 8 = **200 pixel**, necessari per far subentrare lo Screen 2 allo Screen 1, perché 3 bit consentono un massimo di 8 combinazioni (numerate da **0** a **7**) e consentirebbero una traslazione del video di soli 8 pixel. Si rende quindi necessario compiere le seguenti operazioni, nell'ordine:

➪ impostare il modo 24 righe, per evitare sfarfallii.

➪ inserire progressivamente, nei 3 bit del registro di scroll verticale, valori da 0 a 7.

➪ inserire il valore 0 nei 3 bit suddetti e **subito dopo** spostare il video in basso di una riga (8 bit) semplicemente ricopiando i dati dell'area video ($0400 - $07E8) **40** byte **dopo** (facendo attenzione a **non** invadere l'area da $0800 in poi, che contiene i dati per i programmi in Basic!) e sostituendo la prima riga, che ora è il doppione della seconda, con l'ultima riga dello Screen 2.

➪ ripetere dal punto 2 finchè non si vedrà più lo Screen 1.

Ogni dato letto dalla tabella, essendo un byte, pone a disposizione 8 bit, di cui 3 (bit 0 - 1 - 2) serviranno per comunicare al registro di scroll verticale di quanti pixel (min 0, max 7) deve essere traslato in basso il contenuto del video; i restanti 5 bit serviranno a riempire correttamente i puntatori per trasferire le porzioni di schermo.

La manipolazione dei **5** bit può sembrare abbastanza complessa a prima vista, ma deriva da un attento calcolo che ora illustriamo.

Dovendo visualizzare sullo schermo una parte dello Screen 1 ed una dello Screen 2, ed essendo le aree che accolgono i dati dei loro relativi codici video poste una dopo l'altra in memoria, necessitiamo di un puntatore formato da 2 byte, in grado di assumere valori da 0 a 1000.

Questo permette di far corrispondere al valore **0** l'inizio dello Screen 1 (che in memoria è posto dopo lo Screen 2) e al valore 1000 l'inizio (1000 byte prima) dello Screen 2; in tal modo, usando un solo ciclo di trasferimento, si visualizzano sul monitor parti contigue sia di uno Screen che dell'altro. Avendo quindi a disposizione 5 bit, possiamo usufruire di valori compresi tra **8** e **248**, sempre multipli di 8.

Interessa, però, muoverci all'interno dei 1000 byte di 40 byte per volta in quanto ogni riga è formata da 40 caratteri; quindi possiamo **moltiplicare per 5** il valore letto dalla tabella e privato dei primi 3 bit (tramite un AND #$F8) in modo da compiere passi di 8 * 5 = 40 caratteri per volta. Il limite massimo, tuttavia, eccede il valore 1000, concesso, dal momento che 248 * 5 = 1240: basterà assumere, come valore base, 48 ($30) e non 8; infatti...

1240 - 1000 = 240
240 / 5 = 48

La tecnica adottata sarà quella di sottrarre da un tetto fisso (che *teoricamente* sarebbe l'inizio dello Screen 1) quantità di byte multipli di 40 e forzare la locazione ottenuta in codesto modo in 2 puntatori che serviranno per il trasferimento del testo.

Se, infatti, il valore base $30 (#48) deve corrispondere all'inizio dello Screen 1, situato in 50550, il tetto massimo deve essere posto 48 * 5 byte dopo, ossia in 50790 ($C666): ecco il motivo per cui nella routine viene forzato il tetto massimo di $C666 nei puntatori $FB / $FC.

Moltiplicare in lm

Diremo ora come effettuare una moltiplicazione per 5 lavorando in linguaggio macchina.

Inizialmente si moltiplica il valore per 4. Ciò si realizza con due istruzioni.

La prima, **ASL**, trasla il valore a sinistra di **2** bit dal momento che trasla l'accumulatore a sinistra di un bit depositando il bit 7 nel Carry e inserendo poi uno 0 nel bit 0.

```
;*********************************
;*  appendice per variante sonora *
;*          written by            *
;*        f.brl jr soft           *
;*  alias: filippo bruno  *roma*  *
;*********************************
;
;---------------------------------
       sid    = $d400
;---------------------------------
       *=$c95e          ;start=51550
       ldx #$19         ;copia dalla mappa i valori
loop   lda mappa,x      ;necessari per il sid
       sta sid,x        ;
       dex              ;
       bpl loop         ;per ultimo, ovvio, inserisce
       lda #$41         ;nei registri di forma d'onda
       sta sid+4        ;delle 3 voci il valore
       sta sid+11       ;$15 (=21) corrispondente
       sta sid+18       ;all' onda triangolare ad anello
       rts              ;torna
;---------------------------------
mappa  .byte $33,2,0,3,0,0,$f0,$36,2,0,4,0,0,$c0,$39,2
       .byte 0,6,0,0,$40,0,$1e,$f3,$4f
;---------------------------------
       tay              ;'salva' la a in y
       sta sid          ;mette la a nel registro
       clc              ;della parte bassa della
       adc #$03         ;frequenza di ognuna delle 3 voci
       sta sid+7        ;sfasata pero' di 3 unita'
       adc #$03         ;per ottenere dei battimenti
       sta sid+14       ;
       sec              ;sottrae 6 ad a per farla
       sbc #$06         ;tornare come prima
       ldx #$00         ;
       rts              ;torna
;---------------------------------
       .end             ;fine del prg
```

La seconda istruzione, **ROL**, trasla la locazione specificata dopo l'istruzione stessa a sinistra di un bit inserendo nel bit 0 il Carry, il bit 7 dell'accumulatore, e depositando successivamente il bit 7 della locazione nel Carry.

Infine, per completare la moltiplicazione per 5, ci si limita ad aggiungere la quantità iniziale al valore appena trovato.

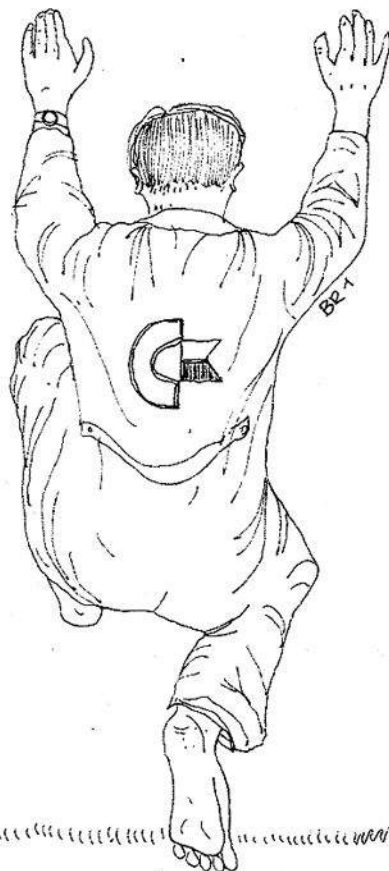
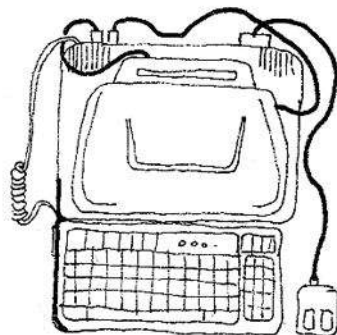
Per quanto riguarda la **variazione sonora**, questa non fa altro che forzare il valore letto dalla tabella nei registri delle frequenze da emettere del SID, inizializzato in precedenza.

Una trattazione più specifica dei parametri inseriti nel SID, come la tecnica per ottenere il fenomeno del *battimento*, o una forma d'onda ad *anello*, esula dallo scopo del presente articolo.

Vi consigliamo, infine, di cambiare sperimentalmente i valori della tabella dello scrolling e di osservarne gli effetti: potrete, ad esempio, cambiarli in modo tale che il rimbalzo avvenga più velocemente e con un numero maggiore di rimbalzi, diminuendo il numero di byte utilizzati per ciascun rimbalzo e aumentando la differenza tra un valore e l'altro.

```
10 rem rimbalzo dallo schermo
11 rem by filippo bruno - roma
12 :
100 lo=49152:printchr$(147)"writing lm...";
110 fori=0to141:readx:k=k+x:pokelo+i,x:next
120 readx:ifx<>kthenprint:print"errore nei data lm!!!":printk:stop
125 print" ok":k=0
130 lo=49294:print:print"writing tabella...";
140 fori=0to255:readx:k=k+x:pokelo+i,x:next
150 readx:ifx<>kthenprint:print"errore nei data tabella!!!":printk:stop
160 print" ok":k=0
169 :
170 rem ------------------------------
171 rem continuare a copiare dalla riga
172 rem 300 se non si vuole aggiungere
173 rem l'appendice con la variazione
174 rem sonora.
175 rem ------------------------------
176 :
```

```
180 lo=51550:print:print"writing appendice...";
190 fori=0to68:readx:k=k+x:pokelo+i,x:next
200 readx:ifx<>kthenprint:print"errore nei data appendice!!!":printk:stop
210 print" ok"
220 lo=49152:pokelo,32:pokelo+1,142:pokelo+2,201:rem --> $c000 jsr $c98e
230 print:print:gosub800
300 printchr$(147)"dopo che compare il punto di domanda"
305 print:print"generato dall' input,cancella lo schermo"
310 print"e riempilo come piu' ti piace (andra' a"
315 print:print"formare lo screen 2)."
320 print:print"tieni presente che la prima riga non"
325 print:print"verra' mai visualizzata....."
330 print:print:gosub800
340 inputa$
350 lo=49550:vi=1024:forx=0to999:pokelo+x,peek(vi+x):next
360 printchr$(147)"controlla la riuscita dell'effetto":print:print:print
370 print"questo e' lo screen 1":print:print:gosub800
380 rem --------------------------------
381 rem continuare a copiare dalla riga
382 rem 400 se non si vuole aggiungere
383 rem l'appendice con la variazione
384 rem sonora.
385 rem --------------------------------
390 sys51550:rem attiva sid
400 sys49257:rem rimbalzo!
410 gosub810:printchr$(147)
420 poke53265,27:rem ripristina 25 righe
430 print"vuoi:":print:print"1 - memorizzare un altro screen 2"
435 print:print"2 - salvare l' effetto su disco"
440 geta$:ifa$=""then440
445 ifa$="1"then300
450 ifa$="2"then460
455 goto440
460 printchr$(147)"inserisci un disco con almeno 6 blocchi":rem 10 blocchi
465 print:print"liberi...":print:print:print:gosub800:rem se c'e' l'appendice
470 open15,8,15,"i0"
480 a$="":printchr$(147)"nome del file (max 16 car.)":inputa$
490 iflen(a$)>16then480
500 open8,8,8,a$+",p,w"
510 print#8,chr$(0);:print#8,chr$(192);:rem pgr parte da $c000 (49152)
520 forx=49152to50549:print#8,chr$(peek(x));:next
529 :
530 rem --------------------------------
531 rem continuare a copiare dalla riga
532 rem 560 se non si vuole aggiungere
533 rem l'appendice con la variazione
534 rem sonora.
535 rem --------------------------------
536 :
540 forx=0to999:print#8,chr$(32);:next
550 forx=51550to51618:print#8,chr$(peek(x));:next
560 close8
570 input#15,a,a$,b,c:print:printa,a$,b,c:print#15,"i0":close15
580 forx=0to24:poke54272+x,0:next:rem resetta tutto il sid
590 end
800 print"premi un tasto per continuare"
810 poke198,0:wait198,1:poke198,0:return
997 rem --------------------------------
998 rem  data prg lm ($c000 -> $c08d)
999 rem --------------------------------
1000 data 168,162,000,134,252,041,007
1100 data 009,016,141,017,208,152,041
1200 data 248,133,251,010,038,252,010
1300 data 038,252,024,101,251,133,251
1400 data 165,252,105,000,133,252,056
1500 data 169,102,229,251,133,251,169
1600 data 198,229,252,133,252,169,000
1700 data 133,253,169,004,133,254,160
1800 data 000,177,251,145,253,200,208
1900 data 249,230,252,230,254,165,254
```

```
2000 data 201,008,208,239,096,162,000
2100 data 189,000,004,157,118,197,189
2200 data 000,005,157,118,198,189,000
2300 data 006,157,118,199,189,232,006
2400 data 157,094,200,232,208,229,096
2500 data 032,075,192,120,169,000,133
2600 data 002,166,002,189,142,192,162
2700 data 000,236,018,208,208,251,172
2800 data 017,208.016,246,032,000,192
2900 data 230,002,165,002,208,230,088
3000 data 096,234,20007
3997 rem -----------------------------
3998 rem  data tabella ($c08e -> $c18d)
3999 rem -----------------------------
4100 data 051,051,052,053,055,058,061
4110 data 065,069,074,080,086,092,099
4120 data 106,114,122,131,140,149,158
4130 data 168,178,188,199,209,220,231
4140 data 242,240,230,219,208,198,187
4150 data 177,167,157,148,139,130,121
4160 data 113,105,098,091,085,079,074
4170 data 069,065,061,058,055,053,052
4180 data 051,051,051,052,053,056,058
4190 data 062,065,070,075,080,086,093
4200 data 100,107,115,123,132,141,150
4210 data 159,169,179,190,200,211,221
4220 data 232,243,241,232,223,215,207
4230 data 198,190,182,174,167,159,152
4240 data 146,139,133,127,122,117,112
4250 data 108,104,101,098,095,093,092
4260 data 090,090,090,090,091,092,094
4270 data 096,099,102,106,110,114,119
4280 data 124,130,136,142,149,155,163
4290 data 170,178,186,194,202,210,219
4300 data 227,236,245,241,235,229,222
4310 data 216,210,204,198,192,186,181
4320 data 175,170,165,161,157,152,149
4330 data 145,142,139,137,135,133,131
4340 data 130,129,129,129,129,130,131
4350 data 132,134,136,138,141,144,148
4360 data 151,155,159,164,169,174,179
4370 data 184,190,196,201,208,214,220
4380 data 225,233,239,246,243,238,234
4390 data 230,226,221,217,213,210,206
4400 data 202,199,195,192,189,186,184
4410 data 181,179,177,175,173,172,171
4420 data 170,169,168,168,168,168,169
4430 data 170,170,172,173,175,176,179
4440 data 181,183,186,189,192,195,198
4450 data 202,205,209,213,217,221,227
4460 data 234,238,243,247,39612
4996 rem -----------------------------
4997 rem data appendice ($c95e -> $c9a2)
4998 rem (non copiare se non interessa)
4999 rem -----------------------------
5100 data 162,025,189,117,201,157,000
5110 data 212,202,016,247,169,021,141
5120 data 004,212,141,011,212,141,018
5130 data 212,096,051,002,000,003,000
5140 data 000,240,054,002,000,004,000
5150 data 000,192,057,002,000,006,000
5160 data 000,064,000,030,243,079,168
5170 data 141,000,212,024,105,003,141
5180 data 007,212,105,003,141,014,212
5190 data 056,233,006,162,000,096,5976
5200 end

ready.
```

I PICCOLI COMMODORE SONO CRESCIUTI

✔ Amiga | ❑ Principianti | ❑ Esperti | ✔ Tutti | ✔ HELP

✔ Ms - Dos
❑ Recensioni
❑ Hardware
❑ Software
❑ Applicazioni
❑ Programmazione
❑ Dos
❑ Pascal
❑ C
❑ Basic
❑ Assembly

Un'antica incompatibilità di formato è destinata a scomparire grazie anche al vostro aiuto. Vediamo come.

Cade il muro tra Amiga ed MS - DOS

< di Alessandro de Simone >

< Il nuovo disco Computer Club Disco accontenta tutti i nostri lettori >

Da molto tempo pensavo che i nostri lettori avevano davvero ragione a lamentarsi delle incompatibilità segnalate da affezionati di una "piattaforma" rispetto ad un'altra.

Il problema presentava, e presenta tuttora, incompatibilità realmente insormontabili operando a livello di programmi memorizzati in codice macchina (leggi: listati compilati nei codici **68000** oppure **80X86**) magari complicati a causa di riferimenti a formati grafici del tutto incompatibili tra loro.

Grazie alla notevole diffusione dei computer, da una parte, ed alla necessità di utilizzare linguaggi di programmazione evoluti che si riferissero ad una sintassi simile, se non addirittura eguale, dall'altra, abbiamo comunque assistito alla proliferazione di linguaggi che, considerando il "ceppo" di provenienza, si assomigliano sempre più fra loro.

Ecco quindi che un "**C**" in ambiente Amiga segue ragionamenti del tutto simili a quelli disponibili in ambiente MS - DOS; il **Turbo Pascal** Borland è quasi indistinguibile, per ciò che riguarda la sintassi, dall'analogo (stavo per dire dal Clone...) **Kick Pascal** Amiga.

Solo i linguaggi **Assembler**, per ovvi motivi, rimangono distanti tra loro; e tali resteranno per molto tempo ancora.

Dunque: se un programma è scritto in **C Amiga** perché mai sfruttarlo solo in quest'unico ambiente? Se una procedura in **Quick Basic** sembra soddisfare le esigenze dell'utente, perché mai non apportare le dovute correzioni per farlo girare in **AmigaBasic**? Un intervento, poi, da parte di un affezionato Amigo di **Amos** potrebbe portare migliorie tali da rilanciare la palla all'utente di **Visual Basic**, il neonato super linguaggio della Microsoft; e così via verso miglioramenti, inserimenti di utility, modifiche dei percorsi, aggiunte di nuove opzioni tendenti a sfruttare sempre più le potenzialità di un linguaggio.

Tutto ciò era già possibile, si potrebbe obiettare, da molti anni a questa parte.

E' sufficiente caricare, ad esempio su un Amiga, i programmi sorgenti scritti con un computer MS - DOS; quindi intervenire opportunamente per le correzioni del caso.

Pur se tali interventi erano possibili anche prima d'oggi, mancavano comunque quelle facilitazioni che una disponibilità immediata, tangibile e, soprattutto, reale, consente di offrire alla gran massa degli utenti.

Come era, infatti, prima d'oggi possibile procurarsi programmi "alieni" per modificarli a piacimento? Praticamente solo tramite le banche dati che, però, offrono quasi esclusivamente file oggetto, inalterabili per definizione; a costi, oltretutto, che tendono a scoraggiare simili esperimenti.

I mondi Amiga ed MS - DOS, quindi, pur se ufficialmente collegabili tra loro, non solo restavano separati, ma offrivano fertile terreno per dispute inutili nelle quali sterili polemiche costituivano l'aspetto culturalmente più elevato.

A partire da oggi

Se, dunque, **Computer Club** ha da sempre rappresentato, nel campo dell'editoria informatica, un preciso punto di riferimento per la didattica, non si poteva fare a meno di realizzare un'iniziativa che tendesse ad appianare polemiche; concretizzando, *conditio sine qua non*, un'idea che è venuta via via maturando nel corso degli anni.

Mi riferisco, in particolar modo, allo scambio culturale ormai possibile tra utenti di diverse piattaforme (**Amiga, MS - DOS, Macintosh**) appartenenti a varie fasce (**principianti, esperti, professionisti**) ed interessati a differenti campi di

CRESCI ANCHE TU COL NUOVO

COMPUTER CLUB

utilizzo del computer (**professionali**, **utility**, **grafica** e, perché no, **giochi**).

Di non secondaria importanza, poi, la considerazione secondo cui, nel software di pubblico dominio, sembra avere predominio l'*intellighentia* d'oltreoceano: mi rifiuto di credere che **in Italia** non ci sia nessuno in grado di scrivere programmi decenti in uno dei tanti linguaggi disponibili.

L'importante, comunque, era offrire un prodotto che potesse interessare, allo stesso modo, sia gli utenti Amiga sia quelli MS - DOS (per le altre piattaforme, vedremo...) per facilitarli nella ricerca di obiettivi comuni e procedure universali.

Computer Club Disco

Disponibile mensilmente in edicola (con periodicità, almeno all'inizio, "sfalsata" rispetto alla presente pubblicazione), i nostri lettori troveranno un nuovo periodico, **su disco**, che rappresenta, appunto, un passo avanti per eliminare le inconcepibili barriere tra Amiga ed MS - DOS.

Sullo stesso dischetto, infatti, troveranno posto sia i file specifici per Amiga sia quelli in formato MS - DOS; la tendenza futura, ovviamente, sarà quella di offrire prevalentemente software, e file in genere, **in formato** ASCII e, come tali, facilmente caricabili e modificabili a volontà da entrambi i sistemi.

Il formato di **Computer Club Disco** sarà quello MS - DOS perché, almeno finora, non risulta disponibile una procedura che consenta di leggere file di formato Amiga con un computer MS - DOS.

Inoltre, aspetto di importanza non certo secondaria, abbiamo notato che il formato MS - DOS offre, in fase di duplicazione, un minor numero di copie mal riuscite (e, quindi, inutilizzabili) rispetto alle copie di dischetti in formato Amiga.

Infine sappiamo che **tutti** gli utenti Amiga possiedono almeno una delle tantissime procedure che trasformano il loro computer in un "lettore" di dischi MS - DOS (basta citare **Dos2Dos**).

Per quei pochi utenti che ancora non dovessero disporre di una di queste, nel primo numero di **Computer Club Disco** (di formato, ovviamente, Amiga) abbiamo provveduto ad inserire un programma, di dominio rigorosamente pubblico, che colmava la lacuna.

Verso altri lidi

Come già detto, siamo solo all'inizio. Per ora inseriremo programmi e file non tutti compatibili al 100% per entrambi i sistemi.

L'importante, ne converrete, era comunque iniziare un nuovo discorso che, inevitabilmente, viene ora passato ai nostri lettori:

Anzitutto questi non avranno più alcun limite di spazio nel proporre i propri listati; inoltre possono suggerire, considerando la particolare ottica con cui verrà gestito il dischetto, nuove iniziative il cui unico limite è rappresentato, come sempre, dalla fantasia.

Ed è proprio quest'ultimo tipo di "merce", ormai lo abbiamo capito, che abbonda tra le caratteristiche dei nostri lettori...

✔ Amiga | ❑ Principianti | ❑ Esperti | ✔ Tutti | ✔ HELP

✔ Ms - Dos

❑ Recensioni

❑ Hardware

❑ Software

✔ Applicazioni

❑ Programmazione

❑ Dos

❑ Pascal

❑ C

❑ Basic

❑ Assembly

Tre routine in Basic, per generare numeri casuali, girano allo stesso modo su Amiga, Ms - Dos e perfino sul C/64!

Casuale come un numero

< di Tarizzo Fabrizio >

< Matematica "pratica" >

Spesso, nella realizzazione di videogame o di programmi di simulazione, vi è la necessità di generare valori casuali, per simulare, appunto, situazioni imprevedibili.

I linguaggi di programmazione implementano una funzione utile allo scopo, ma non tutti sanno *come* vengono determinati i valori random.

Scopo di questo articolo è di spiegare come sia possibile generare sequenze di valori distribuiti casualmente in un certo intervallo (tipicamente tra **0** e **1**). Si noti che il problema è formalmente irresolubile perché un calcolatore è un sistema *deterministico;* è cioè sempre possibile stabilire a priori i risultati delle sue operazioni (per nostra fortuna...).

E' per questo che, in realtà, non viene generata una sequenza casuale di valori ma, tramite particolari algoritmi, una sequenza le cui caratteristiche siano il più possibile vicine a quelle di una sequenza casuale.

Sono stati studiati molti algoritmi per risolvere il problema, alcuni validi ed altri un po' meno. Il metodo più usato è quello della **Congruenza Lineare** che, a partire da un valore iniziale detto **Seme**, genera una sequenza pseudocasuale di numeri, definita dalla formula...

```
n(i+1)=(a*n(i)+c) mod m
```

...dove **n(i)** è il generico valore; **n(i+1)** è il valore successivo; **a** è il moltiplicatore; **c** è l'incremento; **m** è il modulo; **mod** rappresenta l'operatore Modulo (cioè il **resto** della divisione).

I coefficienti a, m, c devono essere maggiori o uguali a zero. Il listato pubblicato, sviluppato con sintassi **Basic universale**, genera sequenze pseudocasuali usando tre terne di valori (a, m, c) predefinite con la possibilità, da parte dell'utente, di inserirne una a scelta per eseguire esperimenti.

Vediamo ora qualche esempio per chiarire le idee e per scoprire le caratteristiche delle sequenze generate.

Innanzitutto scopriamo che, per la presenza dell'operatore **mod**, i numeri generati saranno sempre compresi tra **0** ed **m-1**.

Per seguire gli esempi proposti, digitate il programma, lanciatelo e scegliete l'opzione **4** dal menù principale.

Verranno chiesti i valori dei coefficienti, del seme iniziale, se volete valori compresi tra 0 e 1 (per ora rispondete **N**), ed il numero di valori random da generare (inserite qualunque valore maggiore di m, vedremo in seguito perché).

Inserendo, per prova, i seguenti valori...

```
m = 7, a = 3, c = 5, seme = 2
```

...otterrete la sequenza: 2 - 4 - 3 - 0 - 5 - 6 - **2** - **4** - **3** (eccetera).

Si nota subito che sequenza è **periodica**, cioè i valori **si ripetono.**

Questo è un fenomeno che si riscontra con *qualunque* scelta di parametri. Scegliendo, invece, come seme iniziale 5, la sequenza sarà...

5 - 6 - 2 - 4 - 3 - 0 - 5 - 6

... ossia *la stessa sequenza di prima,* ma con inizio da **5** anziché da **2** (cioè con i valori traslati di una certa quantità).

Nella sequenza vista, **m** vale 7, quindi i numeri generati andranno da 0 al 6; tra questi ne manca solo uno: l'1. Scegliendo questo valore, come seme, otteniamo la sequenza: 1 - 1 - 1 - 1...

Modifichiamo ora il parametro **a**, ponendolo uguale ad **1**. Con un seme uguale, ad esempio, a **3**, otterremo la sequenza...

3 - 1 - 6 - 4 - 2 - 0 - 5 - 3 - 1...

...in cui appaiono tutti i numeri compresi tra 0 ed m-1, una volta ciascuno, senza ripetizioni ed è molto difficile, senza conoscerlo, trovare un legame tra un valore ed il successivo.

La sequenza trovata può essere quindi considerata sufficientemente casuale. Qualunque valore sia assegnato al seme iniziale, la sequenza sarà la stessa, pur se traslata di qualche posizione. In que-

```
30 REM Generatore random
50 REM(metodo di Congruenza lineare)
55 REM n (i+1) = (a* n (i) + c)mod m
60 REM Versione universale
91 REM
92 PRINT " Scegliere terna voluta"
93 PRINT " 1) Leormonth-Lewis"
94 PRINT " 2) Knuth": PRINT " 3)
Goodman-Miller"
95 PRINT " 4) Altra terna"
96 INPUT "Scelta "; r
97 IF r < 1 OR r > 4 THEN 96
99 ON r GOSUB 120, 220, 320, 410
100 GOTO 500
110 REM Leormonth-Lewis
120 PRINT "Terna di Leormonth-Lewis"
130 PRINT " m=2^31=2147483648"
140 PRINT " a=2^16+3=65539"
150 PRINT " c=0"
160 a = 65539: c = 0: m = 2 ^ 31
170 RETURN
200 REM Knuth
220 PRINT "Terna di Knuth"
230 PRINT " m=2^31=2147483648": m =
2 ^ 31
240 PRINT " a = int (*10^8) =
314159265": a = 314159265
250 PRINT " c=453806245": c =
453806245
270 RETURN
300 REM Goodman-Miller
320 PRINT "Terna di Goodman-Miller"
330 PRINT " m=2^31-1=2147483647"
340 PRINT " a=7^5=16807"
350 PRINT " c=0"
360 m = 2 ^ 31 - 1: a = 7 ^ 5: c = 0
370 RETURN
410 PRINT "Inserire terna di valori"
420 INPUT "Inserire valore di m"; m
430 INPUT "Inserire valore di a"; a
440 INPUT "Inserire valore di c"; c
450 RETURN
500 INPUT "Seme iniz. sequenza"; n
501 INPUT "ValoriTra 0 e 1(s/n)"; a$
502 IF a$="s" THEN m1= m-1: GOTO 510
503 IF a$ = "n"THEN m1 = 1: GOTO 510
504 GOTO 501
510 INPUT "Numero di valori"; i
520 PRINT n / m1
530 FOR q = 1 TO i - 1
540 p = a * n + c
550 o = p - INT(p / m) * m
560 n = o: PRINT n / m1
570 NEXT
575 a$ = ""
580 INPUT "Ancora (s/n)"; a$
590 IF a$ = "s" THEN RUN
600 IF a$ <> "n" THEN 580
610 END
```

Il listato che genera numeri casuali viene pubblicato nella versione "universale" Basic

sto caso si raggiunge la massima lunghezza possibile della sequenza, ovviamente uguale ad **m**.

Questo perché se, dopo m passaggi, si sono trovati tutti i numeri compresi tra 0 ed m-1, grazie alle caratteristiche della formula, al passaggio (m+1)-esimo si dovrà ripetere un numero già trovato.

Riassumendo

Studiando il metodo della congruenza lineare, abbiamo dunque individuato le seguenti proprietà:

- Le sequenze sono periodiche.
- La lunghezza massima della sequenza è pari al parametro m.
- Particolari valori di altri parametri possono accorciare la sequenza.
- I valori generati sono compresi tra 0 ed m-1.

Per ottenere numeri tra 0 ed 1, come nelle funzioni implementate nei linguaggi più noti, basterà dividere per m-1.

Provate ancora con le terne inserite nello stesso programma pubblicato (ovviamente non chiedete di generare più di m valori) oppure con terne scelte da voi.

Esistono criteri particolari per una scelta corretta dei tre parametri:

- c ed m devono essere primi tra loro, cioè non devono avere divisori comuni.
- i divisori primi di m devono essere divisori di a-1 (il 4 deve essere considerato numero primo).

Le terne più usate, incluse nel listato pubblicato, sono:

- Terna di Leormonth - Lewis:

$m = 2^{31}$
$a = 2^{16}$
3 c = 0

- Terna di Knuth:

$m = 2^{31}$
$a = int (pigreco * 10^8)$
c = 45380624

- Terna di Goodman - Miller:

$M = 2^{31}-1$
$a = 7^5$
c = 0

Nei linguaggi il seme iniziale viene generalmente fissato in modo hardware, prelevandolo da locazioni di memoria usate per altri scopi, tra cui il clock di sistema.

Per concludere, un **indovinello** per i lettori, anche non espertissimi. L'operazione di modulo, che richiede una divisione, viene spesso realizzata anche con un'operazione logica, molto più veloce nell'esecuzione. Questo trucco si può usare solo con le terne di Knuth e di Leormonth - Lewis, o con altre che abbiano "qualcosa" in comune con queste (che cosa? chissà!). Provate a definire l'operazione e a dimostrare la soluzione ottenuta. Per aiutarvi vi suggeriamo di riferirvi ai numeri binari.

Anche su disco

Come intuitivo, anche questi semplici programmi sono inseriti su **Computer Club Disco** di questo mese, nella sezione dedicata al software "interscambiabile" tra Amiga ed MS - DOS. Per esercizio, provate ad implementare le routine proposte in altri linguaggi, nonostante, come già precisato nell'articolo, siano disponibili specifiche funzioni per determinare numeri random.

✔ Amiga	❑ Principianti	❑ Esperti	✔ Studenti	✔ HELP
✔ Ms - Dos				
✔ C/64				
❑ Hardware				
❑ Software				
✔ Matematica				
❑ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
❑ Assembly				

Due interessanti programmi, in versione universale (Amiga, Ms - Dos e perfino C/64!) per risolvere equazioni in modo insolito.

Una procedura per tutte le equazioni

< di Tarizzo Fabrizio >

< Si lancia anche una sfida ai lettori! >

Quest'articolo è dedicato a chi abbia la necessità di risolvere equazioni in modo semplice e, soprattutto, veloce.

Esistono, in verità, molti algoritmi per lo scopo, ma la maggior parte di essi sono complessi da implementare su computer, specie da programmatori principianti.

I due algoritmi proposti, invece, oltre ad essere molto semplici, hanno anche la caratteristica di essere universali, ossia di poter risolvere (quasi) tutti i tipi di equazioni, quindi anche equazioni logaritmiche, trigonometriche, eccetera.

La teoria

I due metodi si basano sul fatto che qualunque equazione può essere scritta nella forma **f(x) = 0**, in cui **f(x)** rappresenta una funzione della variabile indipendente **x**.

Di questa funzione è possibile tracciare, in un sistema di assi cartesiani, un grafico che ne rappresenta l'andamento al variare della variabile indipendente.

E' chiaro che, nei punti in cui il grafico interseca l'asse delle ascisse, risulta verificata la relazione f(x) = 0 dal momento che tali punti corrispondono agli zeri (cioè alle soluzioni) dell'equazione di partenza.

Il primo dei due metodi è detto delle **Secanti**, esemplificato in figura 2. Se consideriamo un intervallo [a, b] in cui la funzione sia definita e, ad esempio, nel punto **A** sia positiva e nel punto **B** negativa (o viceversa), unendo con una retta **r** (detta retta secante) i punti A **[a, f(a)]** e B **[b, f(b)]**, troveremo un punto **C** (0, c) che risulta molto più vicino allo zero della funzione (punto **P**) di quanto non lo siano i punti A e B.

Se ora ripetiamo l'algoritmo sostituendo il punto **D** **[c, f(c)]** al punto B, troveremo un punto **E** ancora più vicino a P. Proseguendo in questo modo raggiungeremo il tanto desiderato zero, anche se con un valore approssimato.

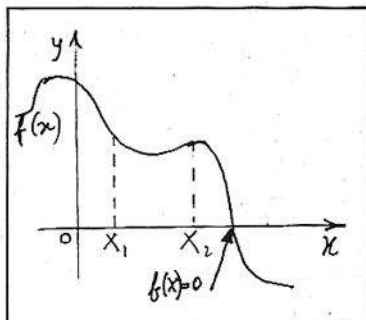



Figura 1

Tangenti

Il secondo metodo è detto delle **Tangenti** (figura 3), del tutto analogo a quello già descritto.

Tracciando la tangente al grafico di f(x) in un punto **A**, troveremo un punto **Ta**, più vicino a zero (**P**) di quanto non lo siano **A'** e **B'**. Ripetendo l'algoritmo nel modo già visto per le secanti, si otterrà lo zero, anche in questo caso leggermente approssimato. Molti avranno capito che i due metodi sono complementari, in quanto uno approssima per difetto e l'altro per eccesso.

L'implementazione

La prima parte è comune ai due programmi, e si compone di una fase dichiarativa, in cui vengono dichiarate le

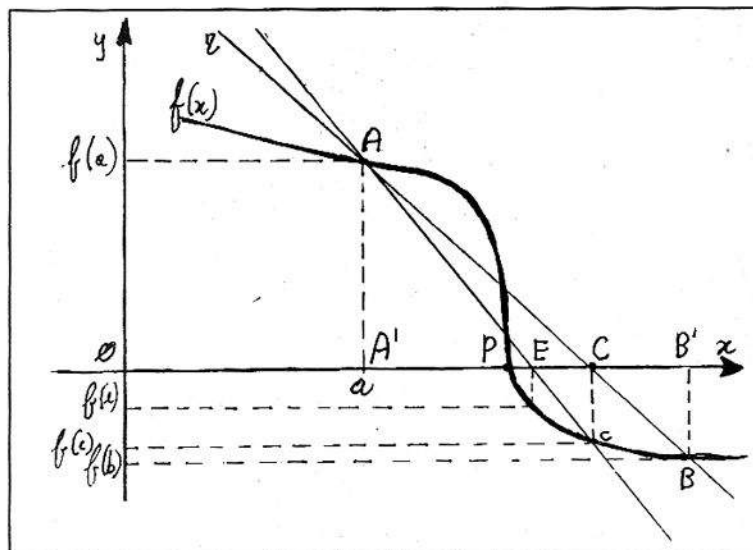



Figura 2

funzioni usate dal programma (linea 100 per il programma secanti e le linee 100, 110, 120 per tangenti).

La funzione DEFinit, in linea 100, corrispondente all'equazione da risolvere, rappresenta una **parabola** non simmetrica rispetto all'asse y. Le tre linee successive richiedono gli **estremi** dell'intervallo di ricerca e la **precisione** del risultato.

Per quanto riguarda il metodo delle secanti sarà sufficiente ricorrere ad alcune formule di geometria analitica.

Risulta necessario tracciare una retta passante per i punti A e B che, come tutte le rette, è individuata dall'equazione **y = mx + n**.

Si determinano i parametri **m** ed **n**, impostando un sistema di due equazioni in due incognite (linee 140 - 150).

Per trovare il punto C (intersezione della retta con l'asse delle ascisse), si risolve l'equazione di primo grado **mx + n = 0** (linea 160).

Le linee 170 e 180 sostituiscono al punto A (oppure B) il nuovo punto trovato, mentre la 190 controlla se il risultato è sufficientemente preciso in base al valore di precisione richiesto; in caso contrario ripete il ciclo. L' implementazione del metodo delle tangenti è leggermente più complessa in quanto dobbiamo scomodare l'analisi matematica, in particolare il concetto di derivata di una funzione.

Senza entrare in lunghi (e noiosi) discorsi, vi basti sapere che la derivazione è un'operazione utile per determinare il coefficiente angolare (cioè il parametro m) della retta tangente ad una curva in un certo punto, definita come il **limite**, per **h** tendente a **0**, di **(f (x+h) - f (x)) / h**.

Con *tendente a 0* si intende che **h** deve essere molto piccolo, ma **non** nullo. Se alla funzione così ottenuta applichiamo nuovamente l'operazione di derivazione, perveniamo alla derivata seconda della funzione di partenza.

Le derivate prima e seconda sono definite alle linee 110 - 120. Esistono altre applicazioni delle derivate nello studio della **crescenza** e della **concavità** delle funzioni, ma questo argomento, almeno per i nostri scopi, non è importante.

Nel programma, la derivata prima determina il coefficiente angolare della tangente, mentre con la derivata seconda si individua l'estremo (A oppure B) attraverso cui far passare la tangente: questa infatti deve passare per il punto in cui la funzione ha lo stesso segno della derivata seconda.

In moltissimi casi, infatti, si potrà notare che, facendo passare la tangente per un punto piuttosto che un altro, ci si allontana dallo zero invece di avvicinarsi (punto H, figura 3).

La parte finale del programma è simile a quella del primo listato.

Uso pratico

Dopo aver caricato il programma, bisogna modificare la **linea 100** per

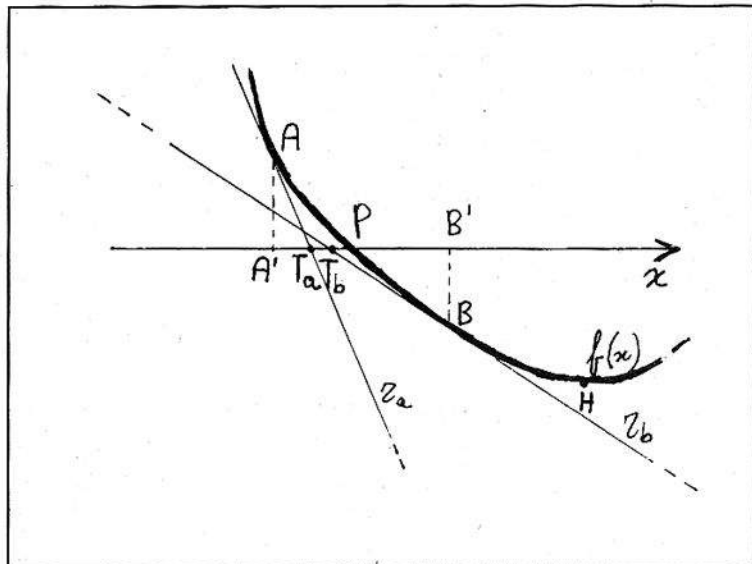



Figura 3

definire la funzione corrispondente all'equazione da risolvere.

Ad esempio, per risolvere l'equazione...

$X^4 - 5 * X^2 - 3 = 0$

...si dovrà modificare la linea come...

```
100 DEF FNF (X) = X^4 -5*X -3
```

Dopo aver dato il Run, basterà inserire i valori dell'intervallo di ricerca e la precisione desiderata. Per evitare di perdere tempo in tentativi inutili, definite a priori gli intervalli da studiare tracciando, almeno a grandi linee, il grafico della funzione, magari con l' aiuto di un programma per grafici matematici.

La parabola presente in riga 100, risolta con i metodi tradizionali, ammette due soluzioni per x = **1** ed x = **3**. Vi suggeriamo di provare un'altra equazione...

Cos(x) - Log(x) + 5 = 0

...che presenta una delle sue soluzioni compresa tra **149** e **150**.

Per quanto riguarda la precisione, generalmente 1e-7 (10^ - 7) è un valore idoneo. A volte è possibile stabilire valori più piccoli, ma in molti casi questi non potranno essere raggiunti a causa del limitato numero di cifre utilizzabili dal computer.

Controindicazioni

Il metodo delle secanti si può usare soltanto se i valori della funzione, ai due estremi dell'intervallo, sono di segno opposto. In caso contrario la secante incontrerebbe l'asse delle ascisse al di fuori dell'intervallo, rendendo impossibile l'applicazione corretta dell'algoritmo. Col metodo delle tangenti l'inconveniente è eliminato, ma alcuni problemi possono insorgere nel caso di funzioni particolari.

Per questo motivo, quindi, si consiglia di studiare attentamente il grafico della funzione prima di decidere il metodo da utilizzare.

```
10 PRINT "** Soluzione di equazioni **"
20 PRINT "** col metodo delle secanti  **"
60 PRINT "** Vers. C/64, Amiga, Ms-dos **"
90 REM Parabola
100 DEF fnf (x) = (1 / 2) * x ^ 2 - 2 * x + 3 / 2
105 PRINT "(L'equazione e' in riga 100)"
110 INPUT "Estremo sinistro "; a
120 INPUT "Estremo destro "; b
130 INPUT "Precisione "; ep
135 REM applicazione metodo secanti
140 m = (fnf(a) - fnf(b)) / (a - b)
150 n = fnf(a) - m * a
160 x = -n / m
170 IF SGN(fnf(x)) = SGN(fnf(a)) THEN a = x: GOTO 190
180 b = x
190 IF ABS(fnf(x)) > ep THEN 140
200 PRINT " x="; x
210 END
```

Miglioramenti

Grazie alla loro semplicità, i listati pubblicati si prestano a personalizzazioni da parte dell'utente.

Ad esempio, si potrebbe modificare il controllo della precisione del risultato in modo da fissare un numero minimo di cifre decimali esatte, invece che applicare il controllo al valore della funzione. Se

provate ad aggiungere una linea che stampi, ad ogni ciclo, il valore ottenuto di X, vedrete che dopo alcuni passaggi questo valore si stabilizza, ed è in base a questa proprietà che si può effettuare il controllo.

Un'altra possibilità è quella di unificare i due programmi, utilizzandoli come subroutine, al fine di risolvere la stessa equazione con entrambi i metodi; ciò allo scopo di confrontare i risultati che saranno comunque sempre molto simili (se non identici) qualunque sia il metodo scelto.

Si suggerisce anche la possibilità di inserire la funzione direttamente da programma invece di modificare, ogni volta, la riga 100.

Su sistemi MS - DOS, e precisamente con il Gw Basic, si può leggere la funzione come se fosse una stringa, salvarla in un file ASCII e usare il comando **Chain Merge** per usarla nell'istruzione DEF di linea 100. Le definizioni delle derivate in linea 110 - 120 determinano valori approssimati, ma chi conosce l'analisi matematica può sostituirle con le derivate vere e proprie.

Ad esempio, le derivate della funzione...

Fnf (x) = Cos(x) - Log(x) + 5

...sono:

Fnf1 (x) = -sin (x) - 1/x

...(derivata prima) e...

Fnf2 (x) = -cos(x) + 1/(x^2)

...derivata seconda)

L'ideale sarebbe inserire anche una routine che tracci il grafico della funzione. Giunti a questo punto, però, è più corretto inserire le routine pubblicate all'interno di procedure matematiche molto più complesse e complete.

Comunque, tentar non nuoce!

Anche su disco

Nonostante la brevità dei listati, questi sono inclusi nel dischetto Computer Club Disco presente questo mese in tutte le edicole.

Grazie alla universalità delle procedure, li troverete nella sezione "comune" ad **Amiga** ed **MS - DOS**.

Coloro che, come suggerito nello stesso articolo, ritengano di aver apportato sostanziali modifiche ai listati pubblicati, sono pregati di telefonare in Redazione (**02/57.60.63.10**) il giovedì pomeriggio per concordarne la pubblicazione che, se meritevole, sarà adeguatamente compensata.

```
10 PRINT "** Soluzione  di   equazioni **"
20 PRINT "** col metodo delle tangenti **"
30 PRINT "**  by   Tarizzo   Fabrizio   **"
40 REM    **  per Computer Club 1991    **
50 REM    **  versione     universale   **
60 REM    **   C/64, Amiga, Ms-dos      **
90 h = .00001
95 REM equazione di una parabola
100 DEF fnf (x) = (1 / 2) * x ^ 2 - 2 * x + 3 / 2
110 DEF fnf1 (x) = (fnf(x + h) - fnf(x)) / h
120 DEF fnf2 (x) = (fnf1(x + h) - fnf1(x)) / h
125 PRINT "(La funzione e' in riga 100)"
130 INPUT "Estremo sinistro "; a
140 INPUT "Estremo destro "; b
150 INPUT "Precisione "; ep
155 REM applicazione metodo tangenti
160 x = b
170 IF SGN(fnf(a)) = SGN(fnf2(a)) THEN x = a
180 m = fnf1(x)
190 n = fnf(x) - m * x
200 x = -n / m
210 IF ABS(fnf(x)) > ep THEN 180
220 PRINT " x="; x
230 END
```

❑ Amiga	❑ Principianti	✔ Esperti	❑ Tutti	✔ HELP
✔ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
❑ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
✔ Assembly				

Facciamo apparire costantemente l'ora corrente mentre, con il nostro Ms - Dos, eseguiamo altri programmi

Un orologio in tempo reale

< di Stefano Degan>

< Il programma è contenuto anche in Computer Club Disco di questo mese >

Il programma **Restime**, scritto in Assembly **80X86**, dopo essersi installato in memoria viene eseguito costantemente grazie alle interruzioni dello stesso timer presente in ogni computer.

Per farlo funzionare in questo modo è stato necessario cambiare il vettore di interruzione **1Ch** facendolo *puntare* ad una nostra procedura residente, preventivamente caricata in memoria.

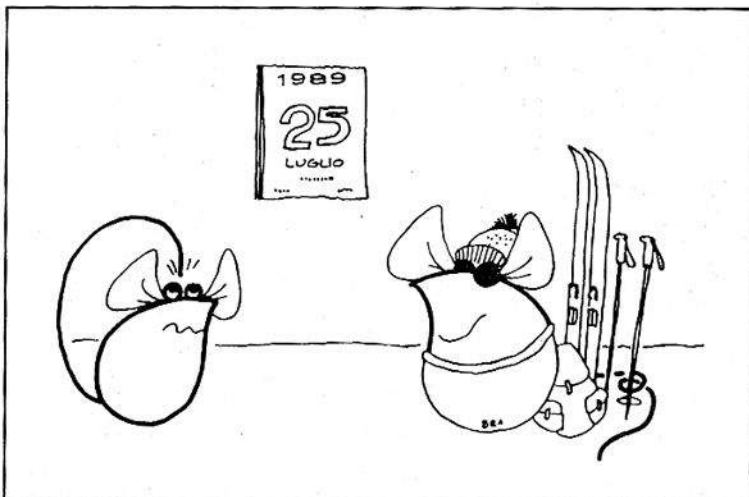
In verità è l'interrupt **08** ad essere eseguito ad ogni pulsazione dell'orologio hardware; se, però, analizziamo la routine relativa a tale interrupt, troviamo, tra le ultime istruzioni, una "misteriosa" **INT 1Ch**, che può essere utilizzata dal programmatore per invocare, alla velocità di circa **18.2** volte al secondo, una routine personalizzata.

E' stato ovviamente necessario rendere il programma **residente** affinché il DOS non sovrapponga un altro programma nella stessa area di memoria in cui risiede il programma in oggetto.

Per motivi di hardware, almeno in alcuni casi, l'esecuzione di INT del DOS **(21h)**, all'interno della procedura residente, può provocare un blocco del sistema; è stato quindi necessario ricalcolare l'intervallo di tempo tenendo ben presente il clock tick del sistema, che, nel programma presentato, è ipotizzato nel numero di **18.2** al secondo. In altre parole vengono conteggiati i segnali di interruzione ed ogni 18.2 di questi viene incrementato l'orario di un secondo.

Probabilità di crash del sistema sono quindi ridottissime. Possiamo assicurare i nostri lettori che il programma è stato fatto girare su un vecchio XT dotato di modesta frequenza di clock, su un 386dx a 25 Mhz e su un notebook 386sx con schermo a cristalli liquidi. In nessuno di questi casi si è verificato nulla di anormale.

La costante visualizzazione della stringa contenente l'orario si effettua attraverso l'interrupt del video **(10h)** che permette la visualizzazione con attributo e in qualsiasi pagina video.

Inoltre è stato indispensabile leggere le *coordinate* del cursore allo scopo di

poterlo riposizionare nella posizione iniziale una volta visualizzato l'orario; questo perché, in caso contrario, il cursore verrebbe sempre posizionato sull'ultimo carattere della stringa rappresentante l'orologio in tempo reale.

I lettori più in gamba possono provare ad inserire la possibilità di disattivare l'orario: per verificare se il programma è attivato, basta infatti controllare che il vettore dell'interrupt **1Ch** punti alla procedura **Orologio**; in tal caso sarà sufficiente rimettere il vettore originale nella tavola dei vettori.

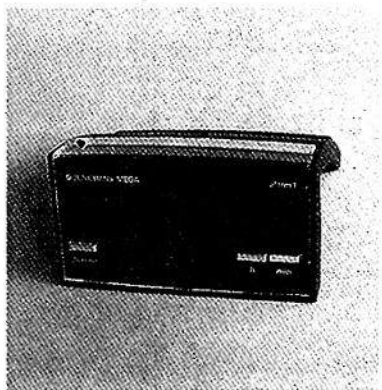
❑❒

Il programma

Il programma è costituito dalla procedura **Orologio** e dalla *macro* **Setta**.

Quest'ultima ha la funzione di trasformare il decimale in ingresso in caratteri ASCII memorizzandoli nella variabile **Strora** usata per immagazzinare l'intera stringa orario destinata ad apparire sul video.

L'operazione di conversione, da numero a stringa, viene effettuata dall'istruzione **Div** che usa, come *dividendo*, il registro **AX**, mentre il *divisore* si trova nel registro che segue l'istruzione; nel nostro caso è il registro **BL**, precedentemente settato a 10.

Come risultato si ottiene il quoziente in **AL** e il resto in **AH**. L'addizione del codice ASCII dello **0** al registro AL e al registro AH fornisce come risultato il codice ASCII della prima e seconda cifra del numero. Per settare il codice nella stringa orario troviamo l'istruzione...

Anche su disco

Il programma qui pubblicato è stato inserito nel dischetto **Computer Club Disco** di questo mese, presente in tutte le edicole.

Restime è memorizzato sia nella versione **sorgente** (`Restime.Asc`) che **compilata** (`Restime.Com`), proprio per venire incontro agli inesperti del linguaggio Assembly che, nonostante ciò, vogliano usare egualmente il listato.

```
mov [DI], registro
```

...in cui le parentesi quadre indicano che il valore contenuto nel **registro** viene settato nella locazione puntata dal registro a sedici bit DI.

Per settare in **DI** (**D**estination **I**ndex) l'indirizzo della stringa *Strora* viene usata l'istruzione...

```
LEA DI, Strora
```

❑❒

Macro

La differenza sostanziale tra una **macro** e una **procedura** sta nel fatto che la macro, durante la fase di compilazione, viene *interamente sostituita* nella posizione del programma in cui viene richiamata, con conseguente occupazione di memoria.

Risulta quindi consigliabile usare la macro quando le istruzioni in essa contenute sono poche.

L'orario viene settato nella variabile a 16 bit **Time** sotto forma di secondi e nella variabile a otto bit **Mezza**. Quest'ultima indica se l'ora è pomeridiana o antimeridiana perché 16 bit non bastano a contenere i secondi di un orario composto di 24 ore: occorre un altro bit.

Continuiamo l'analisi del programma con la procedura residente **Orologio**, in cui vengono inizialmente salvati nello Stack i registri **BP** (Base Pointer) e **DS** (Data Segment) e copiato il registro **CS** (Code Segment) nel **DS** per informare il programma che i dati si trovano nello stesso segmento dei codici.

Quindi viene controllato se si sono verificate 18.2 interruzioni e, in caso affermativo, vengono salvati nello Stack alcuni registri (per sicurezza), incrementato l'orario e settato nella stringa Strora che verrà visualizzata sul video.

Il decimale viene valutato con l'ausilio della variabile **Resto**. Questa variabile, infatti, viene incrementata ogni volta che si verificano 18 interrupt e, quando raggiunge il valore cinque, viene resettata e il programma conteggia un interrupt in più.

La visualizzazione si ottiene con l'ausilio dell'interrupt video. Prima viene acquisita la pagina video attuale, per sapere dove visualizzare la stringa orario, poi viene memorizzata nelle variabili **Riga** e **Colonna** la posizione del cursore.

A questo punto si ha un ciclo di *undici* iterazioni con lo scopo di posizionare il cursore e visualizzare ogni carattere della stringa Strora.

Al termine il cursore viene riposizionato nelle coordinate, precedenti all'esecuzione della procedura, che si trovano memorizzate in Riga e in Colonna.

I registri vengono quindi riportati ai valori iniziali, prelevandoli dallo Stack, e trasmesso il segnale **EOI** di fine interruzione.

Ad eseguire il *prologo* del programma vi è una porzione di codice eseguita una sola volta all'atto dell'esecuzione del programma stesso. Questa, infatti, dirotta l'interrupt **1Ch** alla procedura Orario, preleva l'orario dal timer di sistema tramite la funzione **2Ch** del DOS e ne setta il valore numerico, opportunamente convertito in secondi, nelle variabili **Time** e **Mezza** ed esegue l'interrupt **27H** per rendere la procedura Orario e alcuni dati residenti.

❑❒

Una sfida in Assembly

Come già precisato nel corso dell'articolo, il programma è suscettibile di numerose modifiche ed ampliamenti.

Anzitutto, c'è da precisare che, lanciando **Restime** prima di attivare altri programmi, anche professionali, l'orologio fa bella mostra di sè anteponendosi sempre a tutti i programmi che vengono lanciati successivamente.

Solo con programmi particolarmente complessi (come **Ventura Publisher**) in cui l'installazione degli stessi fa probabilmente piazza pulita di settaggi precedentemente imposti, l'orologio "scompare" per poi riapparire non appena si "esce" dal programma stesso.

Il metodo della scansione del tempo, tuttavia, può lasciare a desiderare perchè *sembra* essere legato all'hardware della macchina. Possono, insomma, verificarsi discrepanze tra l'ora reale e quella visualizzata costantemente in alto sul video, a seconda del computer su cui gira Restime.

Ciò nonostante il programma è decisamente valido e rappresenta una vera e propria miniera di subroutine, idee e procedure da interpretare come suggerimenti per sofisticate sperimentazioni.

In particolare, i lettori più in gamba possono tentare di rendere più breve ed universale la procedura e, soprattutto, inserire un'opportuna routine che sia in grado di **disabilitare la visualizzazione** nel caso in cui, ad esempio, questa si sovrapponga ad una porzione di video che, in un programma professionale, contenga informazioni, per l'utente, non "occultabili".

La possibilità di eliminare l'ora dovrebbe inoltre essere affidata ad una **sequenza di tasti definibile** facilmente dall'utente.

Se, ad esempio, si decidesse di usare i tasti **ALT + Q** per eliminare la visualizzazione, questa stessa sequenza potrebbe entrare in conflitto con un'identica sequenza disponibile nel programma attivato.

La possibilità di "personalizzare" la sequenza di disattivazione dovrebbe essere affidata alla stessa sintassi di lancio, come ad esempio...

`Restime Alt Q`

...oppure all'ultilizzo di un banale file ASCII, facilmente digitabile con un qualsiasi editor di testi, che contenga le opportune informazioni al riguardo. In altre parole, con...

`Restime seq_1`
`Restime seq_2`

...ed altre simili, il programma Restime verrebbe in seguito disabilitato, a seconda di come viene lanciato, dalla sequenza di tasti specificata, rispettivamente, dal contenuto dei file **seq_1** oppure **seq_2.**

Interessante sarebbe, inoltre, determinare la posizione dell'ora stessa (in alto a destra, a sinistra oppure al centro).

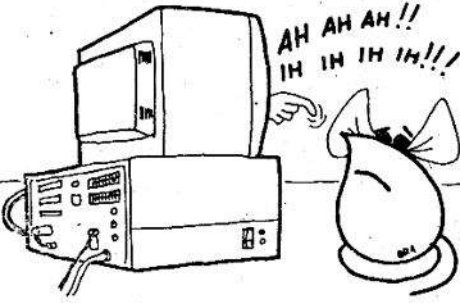
Le idee, come si può notare, non mancano. Tocca a voi rimboccarvi le maniche e... mettervi al lavoro.

Se realizzate qualcosa di valido, contattateci per telefono (**02/57.60.63.10**) il giovedì pomeriggio per concordarne l'eventuale pubblicazione.

```
; Restime.asm scritto in assembly - Stefano Degan 1991
; Visualizza l'orario. Una volta caricato in memoria risulta protetto dal DOS
; MASM/Z restime.asm;     LINK restime.obj     EXE2BIN restime.exe restime.com

; macro che setta nell'indir. puntato da DI il numero Dec in forma decimale.
SETTA       macro Dec
            add di,3               ; Posiziona DI
            xor ah,ah              ; Salva il parametro
            mov al,Dec             ; a 16 bits nello Stack
            mov bl,0ah
            div bl                 ; Divide per 10  AL(quoziente)  AH(resto)
            push ax
            add al,'0'             ; Determina il codice ASCII del resto
            mov [di],al            ; Memorizza dato
            pop ax
            add ah,'0'             ; Determina codice ASCII del quoziente
            mov [di+1],ah          ; Memorizza dato
            endm

; Il segmento dei dati risulta coincidente con il segmento dei codici istruz.
seg_prog    segment public 'code'
            assume cs: seg_prog, ds: seg_prog
            org 100h               ; Predispongo l'inizio del programma
                                   ; all'indirizzo 100H.
program:    jmp inizializza

OROLOGIO    proc far               ; procedura eseguita dal sistema
            push bp                ; 18.2 volte al secondo
            push ds
            push cs                ; Il segmento dati coincide
            pop ds                 ; con quello dei codici

            inc Conta              ; Se sono avvenute 18 interruzioni
```

```
        cmp Conta,18           ; visualizza la nuova ora
        je salto
        jmp exit               ; altrimenti esce e non fa nulla
salto:  mov Conta,0
        inc Resto
        inc Time               ; Incrementa ilválore del Timer
        cmp Resto,5            ; Considera la cifra decimale del 18,2
        jne salto1
        mov Conta,-1
        mov Resto,0

salto1: push ax                ; Salva nello Stack i registri
        push bx                ; modificati dalla procedura.
        push cx
        push dx
        push si
        push di
        push es

        cmp Time,12*60*60      ; Resetta l'orologio se ha oltrepassato
        jne salto2             ; il valore limite 11:59:59,
        mov Time,0             ; in tal caso va a 00:00:00
        mov al,Mezza           ; Nega il bit 0 di Mezza
        xor al,1
        mov Mezza,al

salto2: mov dx,0               ; Determina le ore
        mov ax,Time
        mov bx,3600
        div bx                 ; DX resto, AX quoziente
        mov Ore,al

        mov ax,dx              ; Determina i minuti e i secondi
        mov bl,60
        div bl                 ; AH resto, AL quoziente
        mov Min,al
        mov Sec,ah

        lea di,Strora          ; DI punta all'inizio della stringa Strora

        mov al,'A'             ; Setta AM oppure PM in Strora
        mov ah,Mezza           ; a seconda se l'orario e'
        and ah,00000001b       ; antimeridiano oppure
        je salto3              ; pomeridiano
        mov al,'P'

salto3: mov [di],al
        SETTA Ore              ; Setta le ore in Strora
        SETTA Min              ; Setta i minuti in Strora
        SETTA Sec              ; Setta i secondi in Strora

        mov ah,0fh             ; Acquisisce pagina attuale
        int 10h
        mov NumPag,bh

        mov ah,03              ; Acquisisce posizione del cursore
        int 10h
```

```
        mov Riga,dh
        mov Colonna,dl

        mov Pos,0
Salto4: mov ah,02           ; Posiziona cursore
        mov bh,NumPag       ; nell'attuale pagina video
        mov dh,0            ; in alto a destra
        mov dl,69
        add dl,Pos
        int 10h

        mov ah,09           ; Visualizza i caratteri
        lea di,Strora       ; presenti nella variabile
        xor dx,dx           ; Strora
        mov dl,Pos
        add di,dx
        mov al,[di]
        mov bh,NumPag       ; Pagina video
        mov bl,16h          ; Attributo
        mov cx,01           ; Un solo carattere da visualizzare
        int 10h

        inc Pos
        cmp Pos,11          ; Finche' non sono stati visualizzati
        jne Salto4          ; tutti i caratteri salta

        mov ah,02           ; Riposiziona il cursore
        mov bh,NumPag       ; nella posizione iniziale
        mov dh,Riga
        mov dl,Colonna
        int 10h

        pop es
        pop di              ; Preleva i registri dallo Stack
        pop si
        pop dx
        pop cx
        pop bx
        pop ax

exit:   mov al,00100000b    ; Invia il segnale di EOI all'8259.
        out 20h,al
        pop ds
        pop bp
        iret                ; e ritorna da interrupt.
OROLOGIO endp

Strora  db 'AM:00:00:00'    ; Stringa per memorizzare l'orario
NumPag  db ?                ; Numero pagina attuale
Riga    db ?                ; Riga cursore
Colonna db ?                ; Colonna cursore
Pos     db ?                ; Indice per stampa stringa orario

Time    dw ?                ; Calcolo numerico dell'orario
Ore     db ?                ; Per memorizzare le ore
Min     db ?                ; Per memorizzare i minuti
Sec     db ?                ; Per memorizzare i secondi
```

```
Mezza       db ?                  ; Considera l'orario AM oppure PM (bit 0)
Conta       db ?                  ; Conteggia le interruzioni del timer
Resto       db ?                  ; Viene utilizzata a causa della cifra
                                  ; decimale nel numero delle interruzioni 18,2
resid_end:                        ; eseguite ad ogni secondo

Beep        equ 07h               ; Codice per emettere un segnale sonoro
inizializza:
            push cs               ; Considero i dati nello stesso segmento
            pop ds                ; dei codici istruzione

            push ds               ; Mette DS nello Stack e fa
            mov dx,offset OROLOGIO; puntare il vettore della interr.
            mov ah,25h            ; alla nostra procedura OROLOGIO
            mov al,1ch
            int 21h
            pop ds                ; Ripristina DS

            mov dl,Beep           ; Emissione di un breve suono
            mov ah,02
            int 21h

            mov ah,2ch            ; Legge l'ora
            int 21h
            mov Sec,dh
            mov al,60             ; AX = minuti*60
            mul cl
            push ax

            mov dx,0              ; AX = ore*60*60
            xor ax,ax
            mov al,ch
            mov Mezza,0           ; Indica che e' AM
            cmp al,12             ; Se e' maggiore di 11 allora sottrai 12
            jb jump               ; quindi se e' 12 diventa uno 0
            sub al,12
            mov Mezza,1           ; Indica che e' PM
jump:       mov bx,3600           ; 3600 sono il numero di secondi in un'ora
            mul bx

            pop bx
            add ax,bx             ; Time = Ore*3600+Minuti*60+Secondi
            mov bx,0
            mov bl,Sec
            add ax,bx

            mov Time,ax           ; Mette nella relativa variabile il val.calc.
            mov Resto,0           ; Inizializza le variabili di controllo
            mov Conta,0

            mov dx,offset Resid_end ; Istruzione neccessaria perche'
            int 27h                 ; il programma diventi residente

seg_prog    ends
end         program
```

❑ Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ HELP

✔ Ms - Dos
✔ Recensioni
❑ Hardware
❑ Software
✔ Applicazioni
❑ Programmazione
❑ Dos
❑ Pascal
❑ C
❑ Basic
❑ Assembly

AH AH AH !!
IH IH IH IH!!!

Beh, non esageriamo! Se, però, ci seguite frase dopo frase, risucirete in poco tempo ad intuire le notevoli potenzialità del noto archiviatore

dBase III Plus, conoscerlo in un'ora

< di Alessandro de Simone >

< Se avete dBase III Plus e non sapete da che parte iniziare... >

In ambito MS - DOS sono numerosi i programmi di archiviazione che, realizzati in epoche remote, si sono via via sofisticati e, contemporaneamente, diffusi fino a costituire un vero e proprio standard internazionale.

E' il caso, appunto, del programma **dBase III Plus** che, nonostante la presenza del potentissimo **dBase IV**, ha un numero talmente elevato di estimatori che, di fatto, costringe le software house ad adottarlo come un vero e proprio standard di riferimento.

Molti utenti, soprattutto impiegati di uffici tecnici e commerciali, hanno sul proprio computer (o in rete) la possibilità di usare il potente programma; tuttavia, spaventati dalla considerevole mole del manuale originale di istruzioni, tendono a scoraggiarsi e tralasciano l'opportunità di usarlo, pur se al minimo delle sue potenzialità.

In queste pagine passeremo in rassegna i principali comandi di dBase III Plus in modo da realizzare, con la minima fatica (e, soprattutto, **nel minor tempo possibile**) archivi di *limitata* importanza.

Coloro che, tuttavia, volessero approfondire l'argomento, troveranno una scheda di valutazione su un libro dedicato al programma, abbastanza esauriente, rintracciabile in qualsiasi libreria specializzata.

```
Indice di  C:\DBASE
DBASE     COM     19456
CONFIG    DB         23
HELP      DBS     66560
ASSIST    HLP     17642
DBASE     LD1    138752
DBASE     MSG     12276
DBASE     OVL    272384
DBASEINL  OVL     27648
10 Archivio(i)
```

Figura 1. File indispensabili al funzionamento di dBase III Plus e loro lunghezza espressa in byte.

Iniziare con dBase III Plus

Anzitutto, prima di iniziare, è bene accertarsi di avere a disposizione i file che occorreranno, che sono almeno sei e che si trovano tutti in una stessa directory, di solito chiamata **Dbase** (vedi figura 1). Il file **Dbase.Com** è il programma "principale" incaricato di attivare gli altri segmenti di programma.

Il file **Config.Db** contiene alcuni parametri di configurazione di dBase III Plus, settati al momento dell'installazione.

Il file **Help.Dbs** contiene tutti i messaggi di aiuto formattati in modo particolare e configurati come ipertesto (sono cioè possibili ricerche incrociate durante l'uso di dBase III Plus).

Il file **Assist.Hlp** contiene tutti i messaggi di aiuto che vengono visualizzati quando, attivato il programma di assistenza (chiamato, appunto, **Assistente**), si preme il tasto di aiuto **F1**.

Con Assistente, infatti, l'uso di dBase III Plus viene facilitato grazie alla manipolazione dei menu che guidano l'utente nella gestione del programma. Per attivare l'Assistente è sufficiente premere il tasto **F2**.

I file **Dbase.Ld1**, **Dbaseinl.Ovl** e **Dbase.Ovl** contengono parti del programma e vari messaggi che compaiono durante il suo uso.

Il file **Dbase.Msg** è la raccolta di tutti i messaggi di errore che vengono visualizzati al momento opportuno.

DBase III Plus, per funzionare correttamente, richiede la presenza, nel file **Config.Sys**, dei comandi **Files = 20** e **Buffers = 15**, che noi immaginiamo già settati prima di iniziare. Se non sapete che vuol dire, beh, leggete il manuale del

Campi, record e archivi

Per la gioia di **chi non sa assolutamente nulla di un archivio**, siamo costretti a ripetere cose trite e ritrite. Ci limiteremo, però, ad esporre le definizioni nella forma più breve possibile.

Ogni **archivio** (o *database*), qualunque sia la sua struttura (supponiamo un elenco telefonico), è formato da un certo numero di **record** che, nella similitudine, è rappresentato dal numero di *nominativi.*

Un record, a sua volta, viene suddiviso in **campi**, che altro non sono se non la suddivisione ordinata di informazioni omogenee. Ad esempio, sempre riferendoci all'elenco telefonico, troviamo il campo del *cognome*, quello del *nome*, quello della *via* e quello del numero di *telefono*.

I campi possono essere di diverso tipo: **carattere** (in cui i caratteri usati sono di tipo **alfanumerico,** come quelli di nome, cognome e via); **numerico** (si possono inserire solo caratteri numerici, come i numeri di telefono); **data** (si possono digitare soli gruppi di caratteri che rappresentano, secondo un certo codice ben stabilito, una data. Esempio: 12/03/92 cioè 12 marzo 1992); **logico** (vero, falso); **memo** (una qualsiasi sequenza di qualsiasi tipo di carattere). Non sempre è obbligatorio attenersi alla rigida definizione; ad esempio, si ricorre al tipo numerico solo se avrà senso trattare tali dati in senso matematico. I numeri di telefono, in altre parole, pur essendo *numeri* dovrebbero essere archiviati come *caratteri* sia perchè non avrebbe senso conservare la possibilità di effettuare operazioni matematiche tra numeri di telefono, sia perchè alcune nazioni offrono la possibilità di assegnare numeri di telefono contenenti caratteri alfabetici, sia perchè (motivo principale) digitando un prefisso che inizia con *zero*, questo non verrebbe memorizzato.

Allo stesso modo la memorizzazione di date, in formato diverso da *data*, renderebbero in seguito difficoltose alcune ricerche. Si pensi, ad esempio, all'archiviazione di una biblioteca, con la possibilità di individuare utenti che non restituiscono volumi da molto tempo. Un campo di tipo *data*, in questo caso, consente rapidissime ricerche di tutti coloro che tardano nella restituziuone di libri presi in prestito.

Altro elemento da prendere in considerazione è la **lunghezza** massima da assegnare a ciascun campo perché maggiore è il numero di caratteri assegnati, minore è il numero massimo di record memorizzabili e maggiore è il tempo che occorre per la loro ricerca. Ad esempio, se per il campo cognome assegnassimo solo **8** caratteri, perderemmo la possibilità di memorizzare, per intero, nominativi come *Berlusconi* o *Melegatti* che verrebbero troncati in *Berlusco, Melegatt.* E' ben vero che, anche se troncati, fornirebbero ancora utili elementi per la loro individuazione; tuttavia si preclude, tra le altre cose, la possibilità di stampare **etichette** chiare e precise nel caso si decida di usare l'archivio per la stampa automatica di indirizzi da incollare sulle buste.

L'ordine con cui memorizzare i vari record può essere (ci mancherebbe altro!) del tutto casuale: un programma di dataBase è fatto apposta per ordinarlo, in seguito, in accordo ad una qualsiasi "chiave" di ordinamento.

Create	Crea
Database file	File di database
Format	Formato
View	Relazione
Query	Richiesta
Report	Prospetto
Label	Etichette

Questo è il primo menu da usare quando si desidera creare un nuovo database. Tutti i comandi, in seguito alla loro attivazione, pongono due domande: sul drive da selezionare (A: B: C: eccetera); sul nome del file (digitare il nome del file e premere Return). In tutti i casi, se vi sono file dotati dello stesso nome sul disco / directory indicati, DBIII chiede conferma prima di sovrascrivere.

Set Up	Imposta
Database file	File di database
Format for Screen	Formato schermo
Query	Richiesta
Catalog	Catalogo
View	Relazione
Quit dBase III	Esce da dBase III Plus

Tutti i comandi, in seguito alla loro attivazione, pongono varie domande: sul drive da selezionare (A: B: C: eccetera); sull'elenco dei file già presenti sul disco selezionato (scegliere il file posizionadovisi e premendo Return). In tutti i casi, se non vi sono file specifici sul disco / directory indicati, il comando viene annullato. Nel caso in cui i vari file siano presenti in sottodirectory, e non nella Root (directory principale), bisogna attivare i corrispondenti comandi in modo diretto e non tramite l'Assistente; quest'ultimo si limita a considerare solo i file presenti nella Root. Ad esempio, digitare...

use a:\mio\libri.dbf

...per il database **Libri.dbf** *memorizzatoa ll'interno della directory* **Mio.**

Iniziamo

Siamo quindi pronti per iniziare. Anzitutto procuratevi un dischetto e formattatelo nel drive **a:** in modo da esser sicuri di seguire perfettamente tutte le istruzioni che indicheremo di seguito.

Fatto partire il programma (**Dbase** e tasto **Return**) compare una schermata di copyright. A questo punto l'utente può premere il tasto **F1** (e... perdersi nei meandri dei messaggi di aiuto) oppure iniziare subito a digitare i **comandi** di dBase III Plus.

Siccome noi siamo molto, molto più furbi, premiamo subito il tasto **F2** per far apparire il comodissimo "ambiente" dell'**Assistente**, che offre la possibilità di usare i menu a discesa.

vostro computer: vuol dire che siete proprio agli inizi con un programma di archiviazione, ma con il mondo MS - DOS addirittura!

Questi sono riportati, in queste stesse pagine, sia nella versione **inglese** (a sinistra, nei riquardi) che in quella **italiana** (a destra). Ciò per andare incontro agli utenti che si siano procurati una qualsiasi delle due versioni.

Per passare da un **menu** all'altro si usano i **tasti freccia** sinistra e destra mentre per selezionare una delle **voci** dei menu bisogna premere i tasti di freccia in alto e in basso.

Non fatevi ingannare dal menu (**Set Up** / *Imposta*) che viene reso disponibile non appena si preme F2. Il *primo* menu da attivare, infatti, è bene che sia **Tools** / *Servizi*, dal quale imposteremo, come drive di default (**Set drive**), il drive **a:** in cui dovrebbe esser presente il dischetto appena formattato.

Il *secondo* menu da utilizzare, per creare **il nostro primo archivio** con dBase III Plus, è **Create** / *Crea* di cui indicheremo la prima "voce" (**DataBase File**).

La richiesta del nome del file (un banale: **Primo_Db** è più che valido, almeno in questi nostri primi esperimenti) farà apparire una schermata in cui digiteremo lo schema del nostro archivio. Anzitutto c'è da notare che, premendo ESC, è possibile abbandonare l'ambiente e tornare all'Assistente. Il tasto ESC è di validità universale: premendolo si esce da una condizione indesiderata (caso di errore, menu attivato involontariamente e così via). E' ovvio che, però, uscendo da un determinato ambiente, il lavoro che si stava ivi svolgendo viene perso.

Da notare, inoltre, che parte dello schermo apparso in seguito a **Create** / *Crea* è occupato da un **pro-memoria** (relativo alla gestione dell'ambiente), utile soprattutto per chi è alle prime armi. Premendo **F1** i messaggi scompaiono per lasciare spazio maggiore all'editing (possibilità di scrivere) e viceversa.

Ecco gli altri menu, riportati così come appaiono nelle versioni americana (sinistra) ed italiana (destra), del programma di archiviazione dBase III Plus.

Position	Posiziona
Seek	Ricerca
Locate	Localizza
Continue	Continua
Skip	Salta
Goto Record	Record n.

N.B. Seek è attivo solo du database indicizzati

Update	Aggiorna
Append	Aggiunge
Edit	Edit
Display	Visualizza
Browse	Scorre
Replace	Sostituisce
Delete	Cancella
Recall	Ripristina
Pack	Compatta

Tutti i comandi di Update sono attivabili se è già presente un archivio in memoria.

Retrieve	Recupera
List	Elenca
Display	Visualizza
Report	Prospetto
Label	Etichette
Sum	Somma
Average	Media
Count	Conta

Organize	Organizza
Index	Indice
Sort	Riordino
Copy	Copia

Modify	Modifica
Database file	File di database
Format	Formato
View	Relazione
Query	Richiesta
Report	Prospetto
Label	Etichette

Tools	Servizi
Set drive	Imposta drive
Copy file	Copia file
Directory	Indice disco
Rename	Rinomina
Erase	Elimina
List structure	Strutt. file
Import	Importa
Export	Esporta

Quale archivio

Non uccideteci, per carità: da un punto di vista strettamente didattico si prende come esempio, di solito, la creazione di un'agenda telefonica.

Eviteremo questo supplizio proponendo, tuttavia, un archivio altrettanto squallido e banale: quello della nostra libreria.

Bisognerebbe, a questo punto, fare uno **schema** delle informazioni che vogliamo archiviare, della **lunghezza** di ciascun **campo** e del **tipo** del campo stesso.

Dal momento che non vogliamo assolutamente perder tempo con carta e penna (per tracciare un progetto di massima di database), seguiremo l'istinto riservandoci, anzi, l'opportunità di apportare in seguito modifiche al nostro database.

Ricordiamo che, per passare da una casella all'altra, si preme, a seconda dei casi, i tasti indicati chiaramente nel menu di aiuto che compare (e scompare...) premendo **F1**.

Execute the command	Esegui il comando
Specify scope	Imposta l'intervallo
Construct a filed list	Imposta l'elenco dei campi
Build a search condition	Imposta la condizione FOR
Build a scope condition	Imposta la condizione WHILE

Questo sotto-menu, attivabile solo se è presente un archivio in memoria, facilita la ricerca dei dati "costruendo" il comando un po' alla volta. Viene visualizzato selezionando diverse voci dei menu principali **Update** *(Delete, Recall)*, **Position** *(Locate)*, **Retrieve** *(List, Display, Sum, Average, Count).*

Cognome	Jones	Roth	Nace
Nome	E.	Stephen	Ted
Titolo	dBase III Plus	PostScript	Ventura 2
Prezzo	45000	72000	68000
Data	01/12/87	01/10/90	01/01/89

Figura 2: Dati relativi ai primi tre libri da archiviare

Dunque: abbiamo chiesto di creare *Primo_db*, abbiamo premuto Return in corrispondenza di *Create / DataBase File* (d'ora in poi ci riferiremo prevalentemente alla versione inglese). I primi **due** campi da creare saranno relativi all'autore del libro (COGNOME e NOME), campi di tipo carattere e di lunghezza **15**. Si noti che il cursore, dopo aver indicato la lunghezza (15) del campo, "salta" la casella relativa a **Dec**: questa, infatti, può esser compilata solo nel caso di campi numerici.

Passiamo ora alla definizione del campo **Titolo** (**30** caratteri di tipo... carattere) e del campo **Prezzo** che, di tipo numerico, fisseremo in **6** cifre per garantirci la possibilità (non si sa mai) di indicare libri dal prezzo fino a L. **999999**. Si noti che ora, volendo, potremmo fissare anche il numero di cifre decimali, ma non approfitteremo di tale opportunità (fisseremo quindi **0** cifre decimali).

Inseriamo, infine, la **data** di edizione, di *tipo data*, che viene automaticamente fissata in **8** caratteri.

A questo punto, anche se la definizione dei campi è insufficiente per un archiviuo di vasto respiro, supponiamo di voler concludere (tasti *Control + End*) e di premere il tasto Return per confermare.

Subito dopo comparirà una domanda (Inserimento nuovi record?) dalla cui risposta dipenderà se vogliamo iniziare subito ad inserire i dati relativi alla nostra biblioteca oppure no.

Supponendo di essere esausti(!) diciamo di no e, subito dopo, abbandoniamo addirittura dBase III Plus (*Set Up / Quit*) riservandoci di continuare il lavoro più tardi.

❑❒

Il vero lavoro

E' questo il paragrafo che vi vedrà impegnati per lungo tempo, qualunque sia il database che deciderete di realizzare.

Si tratta, infatti, di digitare tutti i dati relativi ai vari elementi (record) che costituiranno il vostro archivio.

Dunque: caricate dBase III Plus, attivate l'Assistente con il tasto F2 e, dal menu *Set Up / Database*, selezionate il drive **A:** ed il nome del file che ci interessa (il cui schema è stato creato nel paragrafo precedente ed è vuoto, ma ancora per poco).

Compare la domanda (Il file è di tipo indicizzato?) alla quale risponderemo con un brutale **No**. L'indicizzazione dei file, argomento che esula dallo scopo del presente articolo, consente di "legare" più archivi tra di loro.

Sembra che non sia accaduto nulla; invece il file *Primo_db* è stato caricato in memoria ed è pronto per essere elaborato.

Il menu che ora interessa è *Update / Append*, che attiviamo senza indugio. E' ovvio che ora ci riferiremo a libri in nostro possesso, che potrebbero non figurare nella vostra biblioteca.

Digitateli, comunque, per seguirci nei nostri esempi. In seguito, ovviamente, potrete cancellarli ed inserire altri nomi al loro posto.

Non appena selezionate *Append* compare una schermata in cui, in cima, è presente il solito *Help* che indica la funzione dei vari tasti (da eliminare, eventualmente, con **F1**).

Subito dopo compaiono i *cinque* campi, preceduti dalla rispettiva etichetta (Cognome, Nome, Titolo, Prezzo, Data) digitati in fase di impostazione della "maschera". Si noti che la loro lunghezza (15, 15, 30, 6, 8) è riconoscibile dalla diversa lunghezza della **barra nera** posta in corrispondenza di ciascuna di esse.

Digitate, quindi, i dati relativi ai tre libri indicati nella **figura 2**.

Si noti che, se il campo non viene occupato per intero dai caratteri digitati, bisogna andare a capo (cioè portarsi nel campo successivo) premendo il tasto Return; inoltre, completando lo spazio di un campo, un breve segnale acustico viene emesso per avvertire che il cursore si è posizionato automaticamente nella prima cella del campo successivo; ancora, tentando di digitare caratteri alfabetici nei campi *numerico* (prezzo) e *data* viene emesso un segnale di errore; infine, per ciò che riguarda la data, non è necessario premere il carattere di barra inclinata (/) presente nel campo.

Sorgono subito alcuni dubbi: anzitutto, come dobbiamo comportarci se gli autori di un volume sono più di uno? Come digitare i nomi ed i titoli (tutti in maiuscolo, tutti in minuscolo, la prima lettera in maiuscolo e le altre in minuscolo?).

Un terzo dubbio, relativo alla data di pubblicazione del volume (non sempre riportata e non sempre chiara) si risolve, brutalmente, riportando nell'apposito campo la data del primo gennaio (01/01) dell'anno di prima pubblicazione (sempre riportato per legge) oppure di un anno... "orientativo" nel caso in cui sia stata

Ecco il primo menu per chi usa dBase III Plus per la prima volta.

```
Set Up  Create  Update  Position  Retrieve  Organize Modify Tools
          Database file
          Format
          View
          Query
          Report
          Label

ASSIST          ||<A:>||PRIMO_DB              ||Rec: 1/4         ||      ||
Move selec. bar - ↑↓. Select - ←┘. Leave menu - ↔. Help - F1. Exit - Esc
                       Create a database file structure.
```

strappata la pagina in cui viene di solito riportata tale informazione.

Il secondo dubbio, relativo alla digitazione **minuscolo** o **maiuscolo** dei nomi, non è così pignolo come può sembrare. Nella successiva fase di ricerca, dBase III Plus, infatti, si atterrà scrupolosamente a quanto indicato e il nome **Jones**, ad esempio, sarà considerato diverso da **JONES** con le conseguenze che è facile immaginare.

Facciamo notare che, tra le altre pecche del nostro schema, non c'è traccia della **casa editrice** e del numero di **pagine.** Manca, se non bastasse, uno spazio in cui inserire liberamente i nostri commenti.

E' quindi giunto il momento di apportare **cambiamenti** alla nostra struttura di base.

Cambiamenti

Anzitutto registriamo il nostro esiguo file (tasti **Control + End**) e, una volta comparso il confortevole ambiente dell'Assistente, verifichiamo (*Update / Browse*) quanto già digitato: miracolo! in alto sullo schermo compaiono i nomi dei campi e, ben incolonnati, i dati relativi ai tre volumi finora memorizzati. E' addirittura possibile posizionarsi su un campo specifico (tasti freccia in alto, in basso a destra, ed a sinistra) ed apportare eventuali correzioni.

Torniamo, con ESC, all'Assistente ed attiviamo la voce *Modify / Database File.*

N. Record	1	2	3
N_Pagine	380	353	484
Casa_Editr	McGraw Hill	Addison Wesley	Apogeo

Figura 3

Ricompare la struttura su cui abbiamo plasmato il database; possiamo alterare a volontà nomi, tipi di campi e relativa lunghezza di campi già esistenti e/o crearne altri.

Inseriremo, quindi, la casa editrice (tipo carattere, lunghezza 15) limitando il nome a **Casa_Editr** (con dBase III Plus non è possibile inserire nomi di campi di lunghezza maggiore di 10 caratteri, nè spazi bianchi al loro interno) ed il numero di pagine (numerico, 4) inserendolo tra Titolo e Prezzo; infine inseriremo, in fondo, un campo di **tipo memo** dal fantasioso nome di *Note* (fissato automaticamente a 10 caratteri, ma in effetti di lunghezza libera; in fase di memorizzazione dell'archivio si "entrerà" con **Control + PgUp** e si "uscirà" con **Control + W**).

Il nuovo campo del numero di pagine è stato inserito tra Titolo e Prezzo utilizzando **Control + N** (vedi menu di dBase III Plus). In pratica ora i campi sono 8, posti nel seguente ordine: Cognome, Nome, Titolo, N_pagine, Prezzo, Data, Casa_editr, Note.

Concludendo con **Control + End** (e successivo Return di conferma) è ora necessario aggiornare i record già memorizzati. Attiviamo *Update / Edit* e, con i tasti PgUp e PgDn, rintracciamo il primo record. Con i tasti di freccia in alto e in basso posizioniamoci sui campi appena creati ed aggiorniamoli (come in figura 3) record per record.

Dovrebbe ormai esser chiaro perché, nella creazione di un nuovo database, bisogna fare esattamente *il contrario* di come abbiamo fatto noi: nel caso in cui, in seguito, ci si ricorda di inserire altri campi, l'operazione è, sì, possibile, ma richiede l'aggiornamento dell'intero database, pena la sua incompletezza.

Rimane da chiarire ancora, una volta per tutte, se inserire il trattino eventualmente presente nei nomi (McGraw - Hill, Addison - Wesley) oppure no. Prendete la decisione che più vi aggrada e seguitela per sempre.

Vediamo ora come risolvere il problema del maiuscolo e minuscolo.

Maiuscolo e minuscolo

Attiviamo *Create / Format / drive A: / nome: primo_db* e premiamo il tasto Return. Ciò che ora appare è il **Disegnatore**, cioè uno schermo che ci aiuterà nella impostazione di una "maschera" che faciliterà l'utilizzo in fase di immissione dei dati.

Dei quattro menu a discesa che compaiono (**Set Up**, **Modify**, **Options**, **Exit**) selezioniamo *Set Up / Select Database File / Primo_db.Dbf.* Anche in questo caso sembra che non sia successo nulla, ma lo schema del database su cui intendiamo operare è stato caricato in memoria. Ora attiviamo *Set Up / Load Fields* e premiamo Return. Compariranno i nomi degli otto campi precedentemente creati. Ad uno ad uno selezioniamoli posizionandoci sopra con il cursore e premendo Return, finché ogni campo non sia contrassegnato dal simbolo del triangolino (➤). A questo punto premiamo il tasto **F10**, operazione che farà apparire una schermata simile a quella che conosciamo attivando Edit, ma sostanzialmente diversa, e denominata **Lavagna del Disegnatore**.

Con la massima calma, posizioniamo il cursore lampeggiante sulla prima **X** del

Prima impostazione della struttura del database.

```
                                              Bytes remaining:   3926

 CURSOR  <-- -->  |  INSERT      |  DELETE     |  Up a field:    ↑
  Char:    ← →    |  Char:  Ins  |  Char:  Del |  Down a field:  ↓
  Word: Home End  |  Field: ^N   |  Word:  ^Y  |  Exit/Save:   ^End
  Pan:    ^← ^→   |  Help:  F1   |  Field: ^U  |  Abort:        Esc

     Field Name  Type       Width  Dec        Field Name  Type       Width  Dec

   1 COGNOME     Character    15
   2 NOME        Character    15
   3 TITOLO      Character    30
   4 PREZZO      Numeric       6    0
   5 DATA        Date          8

MODIFY STRUCTURE||<A:>||PRIMO_DB              ||Field: 5/5  ||      ||
                           Enter the field name.
Field names begin with a letter and contain letters, digits, underscores
```

Come usare dBase III Plus
di E. Jones
Edizioni McGraw Hill
Pag. 380 L. 45.000

Il volume è suddiviso in 20 capitoli, e contiene anche un'appendice ed un indice analitico.

Inizia dalla definizione di database in generale e descrive, per sommi capi, dBase III Plus. Passa quindi alla progettazione accurata e razionale di un database ed all'attivazione, in particolare, di dBase III Plus. Si esaminano le modifiche che è possibile apportare e la costruzione degli schemi di input, anche complessi.

Si passa quindi alla descrizione dei vari ordinamenti possibili ed alla loro ottimizzazione.

Un intero capitolo è dedicato ai prospetti in cui vengono descritte molto accuratamente le varie fasi da compiere per ottenere output razionali e personalizzati dei dati, eventualmente filtrati in modo opportuno.

Un altro capitolo si occupa dei file di richiesta, in cui viene descritto l'uso delle parentesi e, finalmente, si passa alla programmazione di dBase III Plus che verrà ripresa anche più avanti grazie ad altri due capitoli. Si, perché forse non tutti lo sanno ma **dBase III Plus è un vero e proprio linguaggio di programmazione** che, specifico per i database, consente di realizzare ricerche a dir poco stupefacenti in tempi relativamente ridottissimi.

Le strutture condizionali, la gestione dei file, la realizzazione di relazioni e cataloghi, gli schemi di visualizzazione occupano altrettanti, approfonditi capitoli che, se ben assimilati dal lettore, gli consentiranno di realizzare output realmente professionali.

Il capitolo sull'interfacciamento con altri file ed altri programmi è particolarmente curato e consentirà, soprattutto agli utenti esperti, di realizzare scambi di dati tra word processor, spreadsheet e dBase III Plus.

Non potevano mancare programmi di esempio, i consigli sulla convertibilità di vecchi archivi realizzati con il precedente **dBase II** ed il modo di usare il potente database in **rete locale**.

Un cenno sui programmi di servizio (dUtil III Plus, Quickcode III, Quickreport e compilatori) occupano l'ultimo capitolo.

Impressioni d'uso

Il volume della **McGraw Hill** è davvero completo, chiaro e ricco di esempi numerosi e ben descritti.

Si riferisce alla **edizione italiana di dBase III Plus** e, di conseguenza, le schermate, i messaggi ed i menu riportati fanno riferimento a tale versione.

Chi non sa nulla di archiviazione, e di dBase III Plus in particolare, può intraprendere con fiducia lo studio del potente archiviatore avendo come unica guida il libro di E. Jones.

Gli esempi riportati, da seguire con attenzione e cura, consentono al lettore di impadronirsi con una certa rapidità dei comandi e di sperimentare applicazioni diverse da quelle suggerite nel libro fin dai primissimi capitoli.

Il tentativo di stendere propri programmi, subito dopo aver compreso quelli descritti nel volume, riesce con una certa facilità, anche ad un utente inesperto.

Non c'è che dire: McGraw Hill, con il libro di Jones, mantiene alto il livello qualitativo della collana informatica per cui la casa editrice è giustamente famosa.

campo Cognome (non è possibile, del resto, posizionarsi sulle altre X) e premiamo ancora il tasto **F10**.

Ricompare lo schermo di prima, ma stavolta viene automaticamente evidenziato il menu a discesa *Modify*. In questo c'è una specie di pro-memoria che ricorda il tipo di campo relativo a Cognome (carattere, 15) e, da questo momento, sarà possibile selezionare (voce menu: Action, premere tasto Return) la possibilità, da parte dell'utente del database Primo_db, di modificare a piacimento il contenuto di tale campo *(Action: Edit/Get)* oppure di limitarsi ad osservarne il contenuto del campo stesso senza la possibilità di intervenire *(Action: Display/Say)*. Quest'ultimo caso è comodo quando si intende salvaguardare l'integrità di uno o più campi da parte degli utilizzatori del database.

Ciò che interessa, tuttavia, è la necessità di obbligare l'utente ad usare sempre e solo caratteri maiuscoli nella fase di aggiornamento e/o correzione del database. A tale scopo ci posizioniamo su *Modify / Picture Function* e digitiamo il punto esclamativo (!), comando che provvederà a trasformare immedi-

Seconda impostazione della "struttura" del database.

```
                                                Bytes remaining:   3897

 CURSOR  <-- -->   INSERT        DELETE          Up a field:    ↑
  Char:    ← →     Char:  Ins    Char:   Del     Down a field:  ↓
  Word: Home End   Field: ^N     Word:   ^Y      Exit/Save:     ^End
  Pan:     ^← ^→   Help:  F1     Field:  ^U      Abort:         Esc

    Field Name  Type      Width  Dec       Field Name  Type    Width  Dec

  1 COGNOME     Character   15
  2 NOME        Character   15
  3 TITOLO      Character   30
  4 N_PAGINE    Numeric      4    0
  5 PREZZO      Numeric      6    0
  6 DATA        Date         8
  7 CASA_EDITR  Character   15
  8 NOTE        Memo        10

MODIFY STRUCTURE||<A:>||PRIMO_DB        ||Field: 1/8        ||     ||
                         Enter the field name.
Field names begin with a letter and contain letters, digits, underscores
```

```
CURSOR  <-- -->     |        UP    DOWN  |  DELETE       | Insert Mode:   Ins
 Char:    ←  →      | Field:  ↑     ↓    | Char:   Del   | Exit/Save:    ^End
 Word:  Home End    | Page:  PgUp  PgDn  | Field:  ^Y    | Abort:         Esc
                    | Help:   F1         | Record: ^U    | Memo:        ^Home

COGNOME
NOME
TITOLO
N_PAGINE
PREZZO
DATA
CASA_EDITR
NOTE         memo

APPEND          ||<A:>||PRIMO_DB              ||Rec: EOF/4        ||        ||
```

dBase III Plus durante l'attivazione della funzione Append.

atamente in maiuscolo eventuali caratteri digitati in minuscolo da parte dell'utente in fase di editing. E' opportuno, però modificare anche la maschera *(Modify / Picture Template)* che consente all'utente, per ciascun carattere del campo interessato, di digitare un certo tipo di tasti e non altri. Ad esempio, in corrispondenza di ciascuna **A** sarà possibile inserire solo caratteri alfabetici; in corrispondenza del carattere di cancelletto (**#**) solo segni numerici; in corrispondenza di X qualsiasi carattere e così via.

Se, in un certo campo lungo 5 caratteri, imponiamo ad esempio la maschera...

AA##X

...l'utente potrà digitarvi, in fase di editing, stringhe come...

aG32%

FR45(

AA..4

...ma non...

45455

r5t67

...e simili, perchè non in regola con la maschera impostata.

Tale opportunità sarà preziosissima per impedire errori di digitazione in fase di aggiornamento o correzione degli archivi.

Nel nostro caso specifico imporremo l'impossibilità di far accettare caratteri in minuscolo *(Modify / Picture Function: !)* e caratteri diversi da quelli alfabetici *(Modify / Picture Template: AAAAAAAAAAAAAAA)* assicurandoci che il numero di "**A**" sia **15** come la lunghezza del campo stesso.

Schermata che appare con la funzione Pictute Function in "ambiente" Disegnatore.

```
Set Up              Modify                 Options            Exit   10:06:09

            Screen Field Definition
             Action  : Edit/GET
             Source:   PRIMO_DB
             Content:  COGNOME
             Type     : Character
             Width:      15
             Decimal:
                                          Character Input Functions
            Picture Function:
            Picture Template:             !   convert to uppercase
            Range:                        A   display only alpha chars
                                          D   American mm/dd/yy date
                                          E   European dd/mm/yy date
                                          S   horizontal scrolling
 Function value▸!                         R   insert <other> char
                                              don't overwrite it

MODIFY SCREEN    ||<A:>||A:PRIMO_DB.SCR        ||Opt: 5/6  _||      ||
    Enter one or more function symbols without using quotes. Finish with ←┘.
          Enter a picture function for editing or displaying this field.
```

Da notare che i record precedentemente memorizzati, anche se da questo momento (non dimenticate *Exit / Save*) appariranno tutti in maiuscolo, sono ancora registrati (forse) in minuscolo ed interessati solo da eventuali ricerche in tale formato. Per facilitarci la vita è bene riprenderli, uno per uno *(Update / Append* e tasti *PgUp, PgDn)* e renderli tutti in maiuscolo (terminare con Control + End).

Ancora una volta viene dimostrata l'enorme importanza di **stabilire fin dall'inizio la struttura ottimale** del database, fin nei minimi dettagli.

Con la lavagna del disegnatore presente sul video, è ancora possibile modificare l'aspetto della maschera che apparirà durante l'immissione dei dati.

Posizionate il cursore sulla "**C**" di Cognome e, assicurandosi di avere la modalità inserimento (**Ins**) attiva, premete il tasto Return.

Vedrete che l'intero schema si sposterà in basso di una riga.

Nella riga in cima allo schermo, che ora si è liberata, inserite un messaggio qualsiasi, come "*Maschera di inserimento per l'archiviazione dei miei libri*".

Non soddisfatti, posizionatevi a destra della "**e**" di Cognom**e** ed aggiungete **dell'autore**. Se siete ancora in Insert, noterete che il campo cognome (riempito di 15 **A** maiuscole, per intenderci) si sposta per fare spazio a ciò che digitate.

Se volete che il nome dell'autore compaia alla destra del suo Cognome, posizionate il cursore sulla prima X del **campo** corrispondente a Nome e premete Return. **Subito dopo** posizionate il cursore nel punto esatto in cui volete che, in seguito, compaia l'area destinata al campo del nome (abbiamo stabilito alla destra di Cognome, lo ricordate?) e premete Return: immediatamente vedrete il campo Nome spostarsi nel punto indicato.

Attenzione, però: l'etichetta "Nome" è ancora rimasa al suo posto; niente paura: ridigitatela alla sinistra del campo corrispondente e cancellatela dall'attuale posizione (sotto Cognome).

Sbizzarritevi a spostare messaggi e campi e non dimenticate, alla fine, di premere *F10 / Exit / Save*.

```
Set Up              Modify              Options      Exit  10:08:13 am

           Screen Field Definition
            Action : Edit/GET
            Source:  PRIMO_DB
            Content: COGNOME                  Character Input Symbols
            Type   : Character
            Width:    15                      A     Any alpha character
            Decimal:                          L     Allow T, F, Y, or N
                                              N     Alpha and digits
           Picture Function:                  X     Any character
           Picture Template:                  Y     Allow Y, or N
           Range:                             #     Allow digits, spaces,
                                                    signs, and periods
                                              9     Allow digits and signs
  Picture value►AAAAAAAAAAAAAAA               !     Convert to uppercase
                                              other Overwrite data unless
                                                    @R function is used

MODIFY SCREEN    ||<A:>||A:PRIMO_DB.SCR         ||Opt: 6/6        ||     ||
        Enter a picture template without using quotes. Finish with ←┘.
         Enter a picture template for editing or displaying this field.
```

Schermata che appare attivando Picture Template (in alto) e schermata visualizzabile rendendo operativa (in basso) la funzione Retrieve per la ricerca e la gestione del database.

```
Set Up  Create  Update  Position  Retrieve  Organize Modify Tools  10:13:33

                                  List
 COGNOME                          Display   Execute the command
 NOME                             Report    Specify scope
 TITOLO                           Label     Construct a field list
 N_PAGINE                                   Build a search condition
 PREZZO                           Sum       Build a scope condition
 DATA                             Average
 CASA_EDITR                       Count
 NOTE

 Field Name          Type     Width  Decimal    =  Equal To
                                                <= Less Than or Equal To
 PRIMO_DB->COGNOME Character  15                <  Less Than
                                                >  Greater Than
                                                >= Greater Than or Equal To
                                                <> Not Equal To
```

Ricerche

Un database non è tale se non consente ricerche di vario tipo. Con dBase III Plus potete individuare record dotati di caratteristiche particolari, determinarne il numero e così via. In queste pagine ci limiteremo alla descrizione di alcune delle funzioni offerte dall'Assistente.

```
Set Up              Modify              Options      Exit  10:18:46 am
Maschera di inserimento dei dati per l'archiviazione dei miei libri

Cogmome dell'autore AAAAAAAAAAAAAAA  Nome  AAAAAAAAAAAAAAA

Titolo          XXXXXXXXXXXXXXXXXXXXXXXXXX

N. pagine       9999
Prezzo          999999
Data edizione   99/99/99
Casa editrice   XXXXXXXXXXXXXX

Note            MEMO
```

Esempio di "maschera" di Input realizzata in ambiente Lavagna del Disegnatore.

Ritornate, pertanto, in tale ambiente e, dal menu *Retrieve*, selezionate *List*. Questo comando, senza altra opzione (premete *Execute the command* del sub-menu) visualizza l'intero archivio, a partire dal primo record, sulla stampante o solo su video.

Se, invece, costruite il comando un po' per volta (ad esempio: *Retrieve / List / Build a search condition / Cognome / Equal to / JONES / No more conditions / Execute the command*) vedrete apparire **solo** i record che soddisfano alla ricerca impostata per costruzione. Ricordiamo che è fondamentale digitare lettere maiuscole o minuscole, pena la mancata visualizzazione dei record cercati.

Per realizzare ricerche incrociate, **dopo** aver indicato, ad esempio, il cognome, imponete una nuova ricerca selezionando *Combine with And* (oppure Or) indicando la casa editrice oppure il prezzo oppure il numero di pagine.

A mano a mano che imponete nuovi limiti alla ricerca, vedrete, sul rigo in basso del video, che il comando viene un po' per volta costruito finché non lo attiverete con il definitivo comando *Execute the command.*

Per effettuare le ricerche è bene posizionarsi sempre sul primo record *(Position / Goto Record / Top)* perché, in caso contrario, eventuali ricerche vengono effettuate a partire dal record in cui ci si è posizionati per ultimi (il n. record è sempre indicato in basso sullo schermo).

Eccetera

Con un po' di pazienza, e facendo diversi esperimenti, sarà possibile utilizzare dBase III Plus almeno nelle sue peculiarità fondamentali.

Del resto non si può pretendere, in poche paginette, di condensare tutte le istruzioni del più noto programma di archiviazione in ambiente MS - DOS.

Chi lo desidera, pertanto, potrà approfondire le proprie conoscenze di dBase III Plus procurandosi il manuale originale accluso alla confezione del programma, oppure l'**ottimo volume della McGraw Hill,** di cui si parla nel riquadro specifico.

Nel dischetto mensile Computer Club Disco verranno pubblicati, un po' per volta, i "profili" di vari database da utilizzare con dBase III Plus, contenenti tutti i campi che possono essere utili per la gestione ottimale dei database.

Oltre alle maschere, verrano anche pubblicati i programmi di ricerca automatica ad essi relativi.

Vi consigliamo, quindi, di imparare ad usare dBase III Plus, se in vostro possesso, dal momento che è certamente uno dei più diffusi programmi di archiviazione di dati oggi disponibili in ambiente MS - DOS.

✔ Amiga | ❑ Principianti | ❑ Esperti | ✔ Tutti | ✔ HELP

✔ Ms - Dos

❑ Recensioni

❑ Hardware

❑ Software

❑ Applicazioni

❑ Programmazione

❑ Dos

❑ Pascal

❑ C

❑ Basic

❑ Assembly

Che cosa c'è nel primo numero della nuova pubblicazione che dovete affrettarvi ad acquistare in edicola.

Il braccio destro di Computer Club

< di Domenico Pavone >

< Finalmente una pubblicazione su disco per aiutare i nostri lettori >

Si era tanto parlato di novità, ed eccole finalmente al nastro di partenza: benvenuti al primo appuntamento con **Computer Club Disco**!

Di che cosa si tratta lo avrete letto in altra parte della rivista, ma non guasta ribadire che il **nuovo** floppy targato Systems, anche se per la sua prima sortita è **riservato interamente all'utenza Amiga**, si propone di allargare gli (angusti) orizzonti di chi ama smanettare su una tastiera, aprendosi anche ad altre prospettive. Detto in póche parole, già dal prossimo appuntamento... ce ne sarà per tutti, ovvero potranno usufruirne sia gli "amighi" incalliti (che non perderanno un briciolo di spazio, garantito sin d'ora!) che gli "msdossiani" convinti. Per non parlare della folta schiera che sfrutta **entrambi i sistemi**, magari anche solo attraverso sistemi di emulazione.

Il **primo** numero, proprio in funzione delle novità che prenderanno corpo sin dal **prossimo** numero di *Computer Club Disco*, è incentrato anche sull'**interazione tra mondo Amiga e universo Ms-Dos**, consentita (tra gli altri) dal sistema di pubblico dominio/shareware **MSH**.

Clickatene il cassetto e date un'occhiata alle istruzioni che contiene, e, se siete un po' più esperti, esaminatela da Shell o Cli: vi troverete all'interno l'intero archivio distribuito dall'autore, compreso il suo nome.

Nel disco trovate anche un comodo trasformatore di icone (**Changer**), un simpatico game (**Sball**), un caposaldo degli antivirus (**VirusX**), un originale gadget finto-virus (**Nightmare**), un aiuto per settare i colori di schermo più efficace dell'originale Preferences (**SetColors**), nonché una ricca directory **Computer_Club**, fitta di programmi e icone direttamente collegati alla nostra/vostra rivista di sempre.

In ogni directory, per guidarvi in questo primo approccio con il nuovo floppy, è presente una icona **Clickami!**, da usarsi *prima* di ogni altra. Questa icona attiva la lettura di brevi indicazioni sfruttando il noto programma **Muchmore** (presente anch'esso su disco): se non ne conoscete le funzioni, premete il **tasto Help** per far apparire un breve prontuario d'uso.

Dalla directory principale di **Computer Club Disco N. 1** potrete accedere anche alla schermata di introduzione al dischetto, visualizzabile in qualunque momento biclickando su **Intro.grafica**.

Se poi vi annoiate nel silenzio della vostra stanza di lavoro(...), potete anche lanciare in sottofondo il rock che accompagna l'introduzione al disco, mandando in esecuzione **Intro.musica**. Salvo problemi di RAM, il pezzo eseguito può essere lasciato attivo mentre si effettuano altre operazioni sul computer, con libertà di utilizzo del mouse.

La directory Computer Club

Questo spazio su disco, che coglie simbolicamente l'eredità della defunta **Amigazzetta**, costituisce il punto di raccordo tra Computer Club e questa sua naturale propaggine magnetica.

Soprattutto a partire dai **prossimi numeri**, verranno infatti dirottati in questa directory quei listati che, per le loro dimensioni, non troverebbero pratica collocazione nelle pagine della rivista su carta.

Per maggiore comodità, questi listati saranno divulgati su disco pressoché in contemporanea rispetto all'articolo cui fanno riferimento, evitando così lunghe e noiose sessioni di copia, spesso seguite da un classico urlo *ma non funziona!*.

Inoltre, vi troveranno posto i vostri contributi degni di nota che, per un motivo o per l'altro, *non possono essere inseriti* nelle pagine della rivista. Per esempio programmi fine a se stessi, che saranno così inseriti nel vasto giro del pubblico dominio, oppure troppo lunghi e com-

plessi per essere compiutamente descritti su carta.

E' comunque gradito ogni tipo di contributo, non solo programmi: **traduzioni di manuali di software di pubblico dominio**, magari scritti originariamente in aramaico (va beh, anche dall'inglese o tedesco può andar bene...), o ancora **immagini grafiche di *qualunque* tipo** (però tenente conto, per pietà, che molti nostri lettori sono abbastanza minorenni...), **musiche**, **icone**, **archivi**azioni particolari (la storia dei Sumeri su disco, che ve ne pare?), l'intera **Odissea** in ASCII... e chi più ne ha più ne metta (di fantasia, ovvio).

Per cominciare, in questo primo floppy troverete una marea di listati in **Amigabasic**, **Amos Basic** e **Pascal** pubblicati nei numeri scorsi della rivista (l'elenco lo trovate nel depliant cartaceo che accompagna il disco), e una subdirectory che diventerà presto un'abitudine per gli acquirenti di Computer Club Disco, dedicata alle sfide lanciate da queste pagine.

A partire dai prossimi appuntamenti ne saranno proposte anche altre, che riguarderanno più espressamente il contenuto di CCD (occorre specificare di quali iniziali si tratta?).

In questo numero 1 (o vogliamo chiamarlo numero zero?) sono già inserite alcune soluzioni proposte da lettori come voi, che riguardano due sfide lanciate nei numeri **79** e **85** di Computer Club.

Bando alla timidezza, dunque. Anche il vostro nome potrà circolare per le edicole di tutta Italia, inserito in questa directory: fatevi sotto! E non dimenticate che **Computer Club Disco** crescerà con voi, per voi, ma soprattutto grazie alla **vostra attiva partecipazione**.

Una nota tecnica: per rispondere alle sfide, come certo saprete, può essere utilizzato qualunque linguaggio. Troverete infatti soluzioni in DOS, AmigaBasic, Amos oppure Pascal. Tutti i listati possono essere consultati biclickando sulle icone con suffisso **.asc** (oppure **.p** per i sorgenti Pascal).

Per mandare in esecuzione i file in AmigaBasic basterà agire come di consueto sulla relativa icona, e inserire il disco **Extras 1.3** quando richiesto espressamente dai requester di sistema.

Clickando invece sulle icone con suffisso **.amos**, non succederà, ovviamente, nulla. Questi file vanno infatti caricati direttamente dall'editor di Amos Basic sfruttando i suoi comodi file requester, e mandati in esecuzione selezionando Run dal menu dell'interprete.

Il file system msh

L'insieme dei file contenuti nella directory **MSH**, tutti (tranne il manuale inglese) sprovvisti di icona e quindi non accessibili da Workbench, consente di gestire, nei normali drive di Amiga, dischi in formato MS - DOS.

Non trattandosi di un vero e proprio programma, l'installazione di questo sistema presupporrebbe una certa conoscenza dei meccanismi di AmigaDOS, oltre che della lingua inglese, per approfondire i contenuti della manualistica.

Per superare queste difficoltà, quando si lancia Computer Club Disco è accessibile una **procedura di installazione automatica**, molto semplice e intuitiva da usare, che produrrà, alla fine, un disco di sistema già fornito della possibilità di manipolare file in formato MS - DOS.

Tutto ciò che occorre fare per disporne è seguire scrupolosamente le istruzioni fornite direttamente sullo schermo.

La procedura **non** fa parte del normale archivio MSH: è da noi stata elaborata per facilitarvi al massimo le cose, soprattutto se non si è molto esperti: **anche un super principiante non avrà difficoltà a sfruttarla**.

Per rispettare le regole, l'installazione non altera comunque in nessuna loro parte i file di MSH, per cui chi fosse più esperto potrà comunque accedere dalla Shell (o Cli) del proprio disco di sistema alla directory MSH di Computer Club Disco ed eventualmente personalizzare la propria configurazione.

Se si è seguita correttamente la procedura di installazione, come già detto, si disporrà alla fine di un floppy di nome **WB2** da usare per lanciare Amiga.

Accedendo alla Shell di questo floppy, si potrà adoperare il *nome di device* **Msh:** per riferirsi a dischi in formato MS - DOS inseriti nel drive di Amiga prescelto (**Df1:** se lo si possiede, altrimenti **Df0:**).

L'utilità è presto detta: se, per esempio, si vuole trasferire su Amiga un file contenuto in un disco MS - DOS, basterà una normale operazione di copia come...

```
Copy Msh:file.txt Df0:
```

...per trasferire il file di nome `File.txt` da un disco MS - DOS ad uno Amiga.

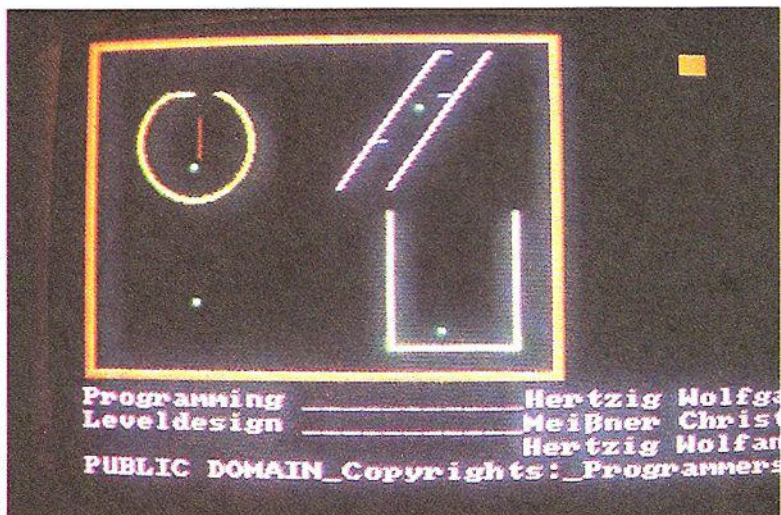
Questo comando, come ovvio, andrà bene solo se si possiede il drive esterno, che funzionerà tanto come **Msh:** che come **Df1:**.

Disponendo del solo drive interno, sarà invece necessario prima un trasferimento in Ram Disk...

```
Copy Msh:file.txt Ram:
```

...con il floppy MS - DOS inserito nel drive. Poi, dopo avere estratto il floppy alieno ed inserito uno Amiga, basterà un banale...

```
Copy Ram:file.txt Df0:
```

...per il definitivo passaggio.

Così come Copy, si potranno usare i più svariati comandi del DOS di Amiga, seguendo la loro normale sintassi, per leggere direttamente file sui dischi MS - DOS (per esempio `Type msh:miofile.txt`), cancellarli (**Delete**), rinominarli (**Rename**), o anche muoversi tra le directory del floppy (**Cd**).

Considerate le differenze di formato, **non** sarà invece possibile adoperare su **Msh:** comandi come **Protect**, **Filenote** e **Install**, specifici di Amiga, nonché **Format**.

E' però ugualmente possibile formattare un disco secondo lo standard MS - DOS. Il disco **WB2** con il nuovo file system installato, dispone infatti di alcuni comandi aggiuntivi, tra i quali **MessyFmt**, preposto appunto alla formattazione.

Questo comando, fornito dallo stesso autore di Msh, richiede però che vengano immessi alcuni parametri prima di agire.

Per vostra comodità, e per un uso più semplice, abbiamo comunque creato uno script già bello e pronto, adoperabile come un normale comando: **`Msformat`**. Basta impartirlo da Shell **senza alcun parametro**, e provvederà a sfruttare automaticamente MessyFmt nel modo migliore, senza alcun vostro intervento.

Se si desidera eliminare il sistema Msh senza resettare il computer, può essere adoperato (sempre da Shell) il comando **Die Msh:**, che provvederà a disinstallare il device dal sistema.

Una nota per chi non avesse confidenza con i Pc compatibili: i nomi di file, in quel sistema, **non** possono avere più di 8 caratteri, eventualmente seguiti da altri 3 come suffisso.

Non lo si dimentichi se si pensa di operare trasferimenti da Amiga a Pc! Per la cronaca, i dischi Ms-Dos inseriti in Msh: risultano visibili anche da Workbench, il che significa che è possibile operare trasferimenti da Amiga a MS - DOS **muovendo le icone come di consueto**. Attenzione, però: poiché i file-icona hanno per Amiga un suffisso **.info**, e l'MS - DOS tollera solo 3 caratteri per i suffissi, si può presentare qualche inconveniente.

L'icona sarà infatti ugualmente visibile da Amiga, ma un normale **List** da ambiente Shell mostrerà come il suffisso sia stato troncato, apparendo come **.inf**. Lo stesso, tra l'altro, avviene con i nomi superiori agli 8 caratteri. Gli stessi file, se poi dovranno essere manipolati da Shell, risponderanno però al loro nome originario, e **non** a quello troncato visibile.

In altre parole, meglio evitare problemi mantenendo nomi di file nell'ambito degli otto caratteri e lasciando perdere le manovre da Workbench, peraltro inutili: le icone, in ambiente Ms-Dos, non servono a niente (almeno quelle di Amiga).

Inoltre, come forse noto, il trasferimento di file di testo è sì possibile, ma occorre tenere presente che questi risulteranno leggibili senza alcun problema solo se redatti in ASCII puro, ovvero **senza** vocali accentate, caratteri **semigrafici**, eccetera. Ma questa non dovrebbe essere una novità per i lettori della rivista, che ha affrontato spesso (e continuerà a farlo, ora più che mai) il tema della conversione ASCII da un formato all'altro.

Per chi mastica un po' di Inglese, è comunque consigliabile una consultazione del manuale originale di Msh, biclickando sulla sua icona nella directory di cui ci stiamo occupando.

Un consiglio: fatevi le ossa con questo sistema (o comunque con uno che svolga compiti similari): vi tornerà utile molto presto.

Non si dimentichi, infine, che il sistema MSH funzionerà **solo** sul disco appositamente installato, **non** è possibile attivarlo direttamente da Computer Club Disco.

E che, se insorgono problemi apparentemente insormontabili, sono sempre disponibili le pagine di **PostAmiga** e la redazione di Computer Club (limitatamente al giovedì).

Changer

Questa utility, anch'essa inserita in Computer Club Disco, consente di cambiare il parametro **Type** ad una qualsiasi icona.

In altre parole: c'è una certa icona che vi piace particolarmente, e che volete associare ad un vostro file?

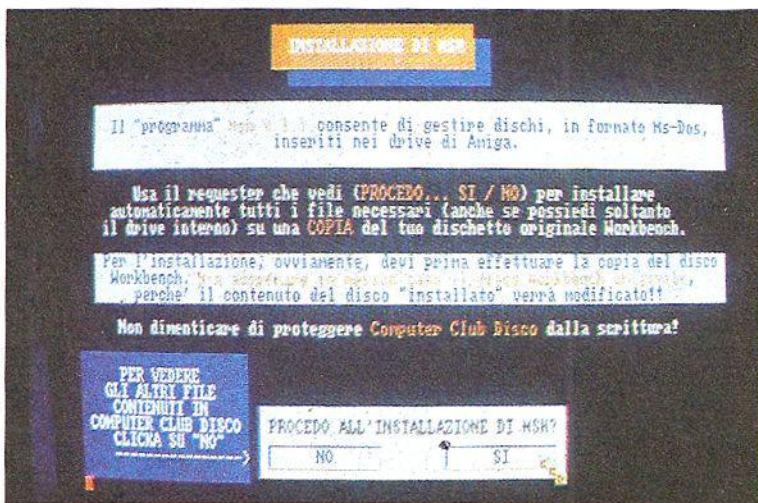
Con Changer potrete farlo, anche se questa è per esempio collegata ad una directory, e quindi non direttamente utilizzabile.

Dopo aver clickato su **Changer_3.0**, basterà prima selezionare il gadget **Load**, quindi digitare nel riquadro che apparirà il nome dell'icona che volete trasformare, debitamente seguito dal suffisso **.info**.

Non dimenticate di adoperare l'eventuale **path** completo del file!

Se, per esempio, volete trasformare il logo della Systems che fa da icona/directory in questo floppy, dovrete digitare: **Computer_Club_Disco:Computer_Club.info**.

A questo punto non resta che attivare col mouse la *leva* di Changer che corrisponde al tipo di icona desiderato (Tool, Project, eccetera), e clickare su **Save**. Nel nuovo riquadro che apparirà, fornite il nome dell'icona prodotta con relativo path e suffisso (per esempio **`Ram:Mia.info`**), premete il Return e il gioco è fatto.

Nella directory **Computer_Club** troverete una raccolta di icone che, con l'aiuto

di Changer, potrete sfruttare in qualunque situazione.

Se avete dubbi o problemi, vale anche qui lo stesso invito prima fatto: armatevi di carta e penna (simbolici: meglio un word processor, non credete?) e rivolgetevi a Postamiga, su Computer Club!

Nightmare

Ecco qualcosa di veramente simpatico, dal nome che la dice lunga. Uno scherzo? Un nuovo virus?

Beh, un po' tutti e due, ma in ogni caso totalmente innocuo, se non per amighi cardiopatici.

Tutto ciò che occorre fare è biclickare sull'icona di Nightmare, o, ancora meglio, inserire qualcosa come `Run Nightmare` (e relativo path) nella startup-sequence di un disco da prestare a qualche amico (eh eh eh).

Sul momento non accadrà nulla, ma continuate a smanettare sul computer per circa 5 minuti, e tenete ben alto il volume del monitor. Il resto lo scoprirete da soli...

Per la cronaca, il fenomeno continuerà a ripetersi **ogni 5 minuti**, fino al reset del computer, senza comunque influire sulle sue normali attività.

Sball

Un tipico esempio di come un game può tenere attaccati al joystick per ore, pur senza disporre di grafica mirabolante, nè di particolari effetti sonori. Provare per credere.

Le regole sono semplicissime: occorre catturare tutti gli elementi **verdi** presenti sullo schermo facendovi passare sopra una pallina, controllata dal **joistick** nelle sue traiettorie rimbalzanti.

Il tutto senza farle toccare alcune sezioni mortali, nel qual caso il game riprende *dall'inizio*.

Già fare uscire la pallina dal contenitore iniziale non è uno scherzo, ma starà a voi trovare la strategia migliore.

Per uscire dal gioco, è sufficiente premere il tasto **Shift di sinistra**.

Setcolors

Quante volte vi è venuta voglia di modificare i colori dello schermo, magari **mentre** state operando in Workbench, ma senza dover ricorrere alle lungaggini di Preferences? Sicuramente molte, vero?

Setcolors, in queste situazioni, può rappresentare lo strumento ideale, in quanto è richiamabile tanto da Shell (o Cli) che da Workbench, in quest'ultimo caso adoperando la sua icona.

Il picolo requester che appare consente di accedere ad una semplice e intuitiva, ma completa, finestra di regolazione della palette, che modificherà in diretta i colori dello schermo.

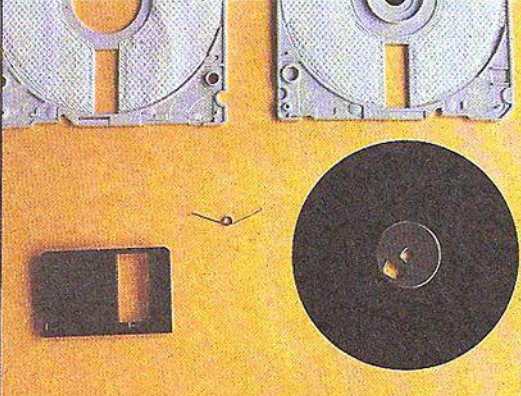
Per di più, è possibile salvare uno o più file di configurazioni predeterminate, richiamabili in qualunque momento sfruttando lo stesso programma. Se poi si vuole rendere definitiva la scelta per un certo dischetto, si può anche salvare i settaggi come system-configuration (opzione **Save Prefs**) come fa il Preference fornito nel disco Workbench, in modo che verranno adoperati ad ogni boot di Amiga con quel floppy.

Il tutto con banali click del mouse debitamente assestati, e un file requester di comodissimo uso.

Se pensate di copiare questo programma in un vostro disco, tenete presente che sarà necessario copiare anche i file **Arp.library** e **Req.library** (presenti nella directory **Libs** di Computer Club Disco) nella omonima directory del floppy che adopererete per il boot.

Virusx 4.01

Si tratta dell'ultima versione disponibile del più noto degli antivirus, già proposto nelle sue precedenti release su Amigazzetta.

IL suo uso è semplicissimo: basta clickarne la sua icona, o inserire un richiamo al programma nella startup-sequence di un proprio dischetto, e VirusX si installerà in memoria **senza disturbare** minimamente le attività del computer.

Con una garanzia, però: controllerà sempre che in RAM non si insedi alcun virus, e contemporaneamente effettuerà la verifica del bootblock di **ogni floppy che verrà inserito** nei drive di Amiga, segnalando l'eventuale presenza di virus o blocchi anomali.

Inutile aggiungere che, previo richiesta, provvede anche a eliminare gli intrusi.

Attivandone la finestra mediante un tradizionale click del mouse con il puntatore al suo interno, si può poi forzare in qualunque momento il check (controllo) dei dischi adoperando il tasto **C**. Allo stesso modo, si potrà visualizzare il contenuto del bootblock di un disco, digitando il numero dell'unità che interessa (**0** per Df0:, **1** per Df1:, eccetera).

Il pericolo è sempre in agguato, meglio non farsi trovare impreparati!

✔ Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ HELP
❑ Ms - Dos				
✔ Recensioni				
❑ Hardware				
❑ Software				
❑ Applicazioni				
❑ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
❑ Assembly				

Power Packer 3.0: l'ultima release del più noto compattatore di file per Amiga

Un super compattatore per Amiga

< di Gregor Samsa > — < Utility per Amiga >

Un posto di rilievo nella softeca di qualunque categoria di computer, è notoriamente occupato dai cosiddetti programmi di compattazione (per gli amici: **cruncher**).

Amiga non fa eccezione, e anzi l'esigenza di ridurre lo spazio fisico occupato dai file in un supporto magnetico si fa sentire forse più che altrove. La complessità del suo sistema operativo richiede infatti la presenza, nel disco di boot (hard o floppy che sia), di una discreta quantità di files.

Se a ciò si aggiunge il dato statistico che vede preponderante l'uso di Amiga basato sui soli floppy disk, l'esigenza di risparmiare quanto più spazio possibile diventa più che evidente.

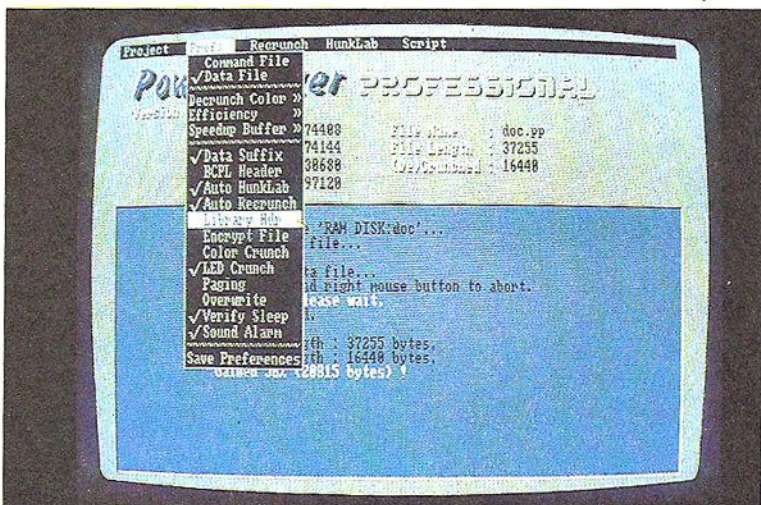
Anche possedendo un hard disk, d'altra parte, non si naviga mai in acque veramente tranquille: fidando nel maggiore spazio a disposizione, al preferito programma di grafica, di solito, se ne accompagna un altro che consenta qualche ritocco in più; e si può tenere solo un word processor, quando spesso può risultare comodo un editor più essenziale? In definitiva: pochi giorni dopo l'installazione, ci si accorge che i 20, 40 o più Megabyte di capienza sono al lumicino.

Per sopperire a queste difficoltà, sin dai primordi di Amiga si son visti proliferare svariati programmi di compattazione, che risultavano spesso difficili da usare, e che non sempre davano il risultato sperato.

Ci stiamo riferendo, infatti, a cruncher in grado, sì, di ridurre le dimensioni di un file, ma di **mantenerne anche l'eseguibilità**, quando necessario.

E, soprattutto in funzione della particolare struttura dei file Amiga (ci si riferisce ai "terribili" **Hunk**), la cosa non sempre era possibile.

Ma già appariva all'orizzonte (ah, poesia...) un programma che spiccava sugli altri: **PowerPacker**.

Sin dalle prime release questo era caratterizzato soprattutto dalla semplicità di approccio, consentito anche ad un ultra principiante, e dalla velocità di esecuzione. Col tempo il programma si è evoluto

in maniera incredibile, sino a giungere alla potentissima attuale versione, la **3.0b**, cambiando nel frattempo anche il suo "status" giuridico: da appartenente al Pubblico Dominio (shareware), è diventato un **programma commerciale**, copyright Uga, che lo diffonde in Europa attraverso una capillare rete di distributori (per l'Italia: L'Agorà, C.so V.Emanuele 15, Milano) cui rivolgersi per l'acquisto.

Peraltro alquanto economico, se si considera un prezzo di **29000 lire** che comprende il programma, alcuni sorgenti, il software di supporto del quale parleremo tra breve, il diritto agli upgrade, ed un manuale su disco (anche **in italiano**!).

Manovre di base

Chi abbia già avuto a che fare con precedenti versioni di PowerPacker (PP), noterà subito, all'avvio del programma, un primo elemento di novità, anche se non così determinante: il *look*.

Abbandonate le precedenti vesti sgargianti in blu e rosso, lo schermo di lavoro del cruncher si presenta ora in più delicate tinte grigio e azzurre, con un'estetica che ricorda le impostazioni del Workbench 2.0.

Ma non è certo questa l'unica novità, come constatabile esaminando più accuratamente le numerosissime opzioni disponibili da menu.

L'operazione di compattazione di un file, si svolge fondamentalmente in **tre** passi: *caricamento* in memoria, *compattazione* (o scompattazione, come vedremo), e suo *salvataggio*.

Le fasi di *input / output* sono gestite da comodi e intuitivi **file requester**, anch'essi in perfetta estetica "a sbalzo", ma prima di adoperarli è opportuno il settaggio preliminare di alcuni parametri del cruncher inclusi nel menu **Prefs**.

Primo tra tutti, quello riguardante il tipo di file da trattare. PowerPacker consente, infatti, di compattare tanto **file eseguibili** (comuni programmi, per intenderci) quanto **file di dati** (testi, grafici, animazioni, eccetera).

Le prime due voci del menu Prefs, appunto, determinano la modalità operativa: **Command** File per i programmi, **Data** File... indovinate per quali?

Se, comunque, non si è impostata la scelta corretta, non accadrà nulla di male. Al momento del Load del file (omonima voce del **menu Project**), PP riconosce se questo è un eseguibile o meno, e in caso di incongruenza con il settaggio scelto segnala la cosa e non procede oltre.

Se, invece, tutto è in regola, carica il file in memoria e procede automaticamente alla compattazione, mostrando a video il procedere delle operazioni, nonché le dimensioni iniziali e finali del file.

Il file compattato viene mantenuto in un buffer, e può a questo punto essere salvato su periferica (opzione **Save** del menu Project) anche più volte (per esempio in più floppy).

Prima di entrare nei dettagli, va detto che l'identica sequenza di operazioni è funzionale anche alla decompattazione.

Dopo il caricamento, PP è in grado di riconoscere se il file scelto è già stato compresso: in questo caso, sempre automaticamente, avvierà la procedura di scompattazione, e potremo (volendolo) affettuare un Save del file tornato alle sue originarie dimensioni.

Per questa fase, PP è in grado di supportare non solo il suo formato, ovvero scompattare file precedentemente compattati con una delle versioni di PowerPacker, ma anche quello di **altri 9 cruncher**, il cui elenco è visibile selezionando il **menu Recrunch**.

Da questo menu, è possibile effettuare proprio l'operazione che si immagina: selezionandolo, dal solito file requester, si potrà caricare un file già compattato che, una volta in memoria, sarà decrunchato e quindi nuovamente ricompattato in accordo con i settaggi al momento impostati. Utile, in pratica, se per esempio si posseggono file compattati con vecchie versioni di PP o con altri cruncher, per ottimizzarli al massimo.

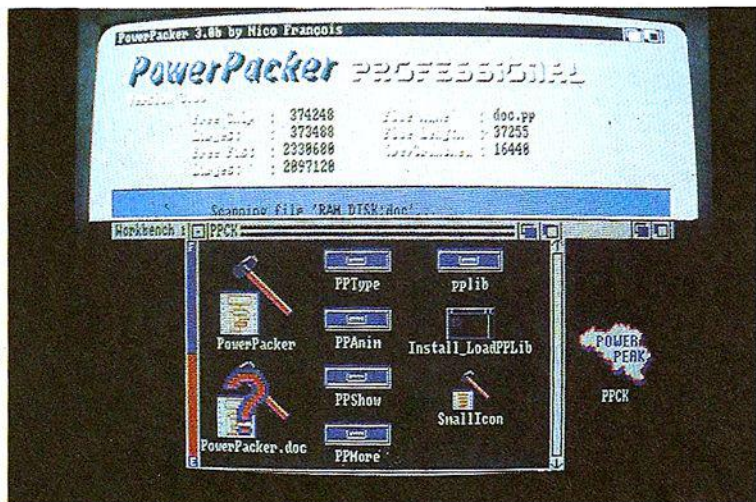
Questo tipo di operazione può anche essere adottata di default, selezionando la voce **Auto Recrunch** dal menu Prefs.

E già che si è in tema di settaggi, vediamo le altre scelte disponibili nel menu appena citato, determinanti sul risultato finale, e che presentano una gradita novità rispetto al passato.

Settaggi e prestazioni

Intanto, si può determinare quale veste estetica assumerà la futura fase di decrunch dei file eseguibili.

Ovvero: una volta compattato, un programma andrà poi mandato in esecuzione come sempre. Solo che, una volta caricato in memoria, il suo header (una porzione di codice aggiunta da PP in testa al programma) provvederà prima a scompattarlo automaticamente.

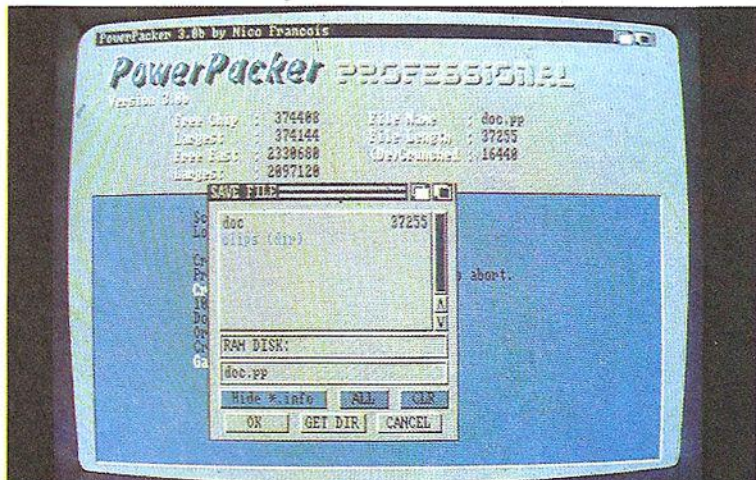
Questa fase, la cui durata è proporzionale alle dimensioni del file, può essere evidenziata da un cambiamento di colori a carico dello sfondo (**opzione De-**

crunch color / Color 0), del primo piano (**color 1**) e del puntatore del mouse (**pointer**), oppure caratterizzata da una specie di "terremoto" dello schermo (scroll).

Scegliendo **None**, si avrà infine solo una pausa, senza effetti particolari.

Maggiore incidenza sul funzionamento di PowerPacker hanno invece le due voci successive, **Efficiency** e **Speed Buffer**, entrambe fornite di submenu con alcune scelte imposte, che influiscono sulla velocità alla quale viene effettuata la compressione, nonché sul suo grado di efficienza. In particolare, maggiore sarà l'efficienza, più lento sarà il processo di compattazione, ma il prodotto finale risulterà di solito più compatto.

Già selezionando l'item **Good** si ottengono buoni risultati, ma **Best** resta quasi sempre la scelta ottimale. Per incrementare la velocità, è invece sufficiente aumentare le dimensioni del buffer adoperato da PP.

La scelta è limitata a dimensioni **Small**, **Medium** e **Large** (un po' come le magliette...). I migliori effetti si ottengono con Large, ma occorre fare un po' i conti con la propria disponibilità di memoria RAM. Il buffer, in questa configurazione, occuperà infatti ben **200 Kb**, ma possedendo 1 Mega di RAM non sorgeranno problemi di sorta.

Per non restare sul vago, e constatare la potenza di PP: operando con floppy disk **un file di testo lungo 83 Kb viene crunchato in 30 secondi con buffer Large ed Efficiency Best. Il file risultante sarà di 38 Kb** (!).

In eguale configurazione, un file eseguibile (di norma meno compattabile) di **89 Kb** può essere portato a **57 Kb** (o meno, dipende dalla struttura del file) in **35 secondi**

Lo stesso, adoperando un buffer Large ed Efficiency Good, impiegherà invece 50 secondi, per dimensioni solo di poco superiori.

Non male, comunque.

Tutte le scelte operate possono poi essere salvate in modo che diventino di **default** ogni volta che si attiva il cruncher.

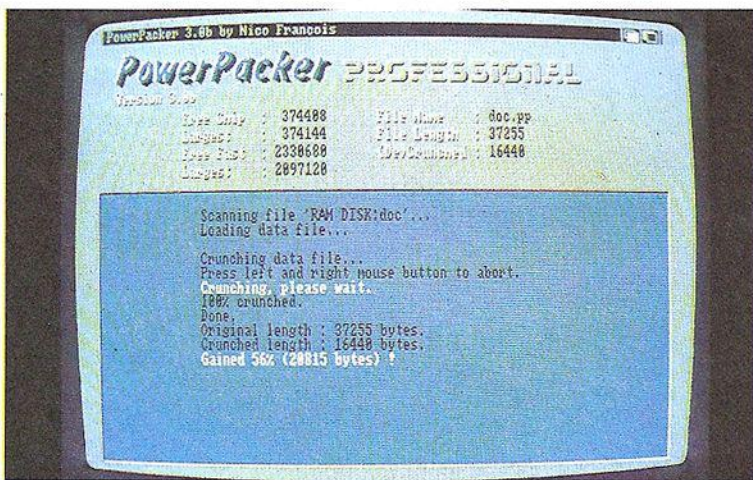
Sempre dal menu Prefs, sono ancora possibili molte opzioni di intuitivo significato, per cui ci limiteremo a citarne due, estremamente interessanti: **Encrypt** e **Library Hdr**.

La prima consente, in pratica, di assegnare una **password** ad un file, senza la quale non sarà possibile accedere allo stesso. Se il file è un eseguibile, al momento del suo lancio (come ovvio dopo il trattamento) apparirà una piccola finestrella che richiederà la parola d'ordine.

Se dopo **3 tentativi** questa non coinciderà con quella assegnata durante il crunch, la finestra scomparirà senza che il programma venga eseguito. Come ovvio la cosa può riguardare anche file di testo o di altro tipo, e la sicurezza è garantita: anche "spulciando" il file con **New Zap** o similari, non si troverà traccia della password, debitamente crittata da PowerPacker.

Novità e tradizione

Library **Hdr**, un'altra delle opzioni presenti nel menu Prefs, costituisce la vera novità della versione 3.0b, colmando una lacuna che faceva talora preferire altri compattatori concorrenti, come **Turbo Imploder**.

Si tratta, in pratica, di un modo per guadagnare ulteriore spazio, diminuendo le dimensioni del già citato header associato ad ogni file eseguibile.

Questo, a seconda dei casi, assume le dimensioni di 600 - 700 byte. Adoperando **Library Hdr**, lo stesso viene ridotto a meno di 100 byte, ma il file, per essere eseguito, chiede che nella directory **Libs** di sistema sia presente il file di nome **PowerPacker.library**.

Inoltre (per gli Amiga con Kickstart 1.2 e 1.3) la startup-sequence deve contenerre un'entry `LoadPPlib`; comando, questo, che a sua volta deve risiedere nella directory **C:**.

Se si è alle prime armi, e non si è in grado di effettuare queste modifiche, con PP è comunque fornito uno script lanciabile anche da Workbench che provvede automaticamente ad organizzare la configurazione.

Il vantaggio dell'opzione è evidente: in un floppy con centinaia di file compattati (per non dire di quanti ne può contenere un hard disk!), basterà sia presente la libreria prima citata, e si risparmieranno svariati altri kilobyte solo con questa mossa.

Ovvio che, in questo caso, i file compattati con l'opzione Library Hdr potranno funzionare **solo se** è presente PowerPacker.library. Tralasciando altre opzioni (tra cui menu **HunkLab**) utili unicamente al programmatore esperto, che saprà utilizzarle senza alcun problema, resta da accennare sulla possibilità offerta da PP di creare script in grado di far risparmiare molto tempo e fatica.

Si tratta di attivare una sorta di registrazione degli avvenimenti (Menu **Script / Start Recording**), dopo di che basterà in pratica eseguire come di consueto tutte le manovre per compattare o scompattare un file e "fermare" la registrazione con **Stop Recording**.

Da questo momento in poi, si potrà far ripetere la sequenza di operazioni al cruncher, senza alcun intervento da parte nostra, selezionando **Execute Script** dallo stesso menu.

Altre utility

Oltre al programma principale, come si è visto di uso molto semplice, Power Packer è accompagnato da una serie di **utility** che ne ampliano e completano la comodità di utilizzo.

Tanto per cominciare, gli amanti del DOS potranno eseguire le operazioni di compattazione o decompattazione direttamente da **Shell** (o Cli) sfruttando due eseguibili non dotati di interfaccia grafica: **Crunch** e **Decrunch**, dall'ovvia funzione.

Sono dotati di molti parametri visualizzabili lanciando i programmi col solo loro nome, e producono (come ovvio) file interamente compatibili PP.

Si rivelano utili per veloci operazioni, senza la necessità di caricare PowerPacker, poi il file da comprimere, eccetera.

Ancora più utili risultano poi i vari tools di gestione diretta dei file di dati compressi. Se, per esempio, si è compattato un testo, comune prassi vorrebbe che, per leggerlo, prima lo si scompatti.

Ebbene, non è necessario: basta utilizzare **PPmore** (fornito, come gli altri di cui si parla, assieme al programma principale), ed il gioco è fatto.

Il programma funziona esattamente come il More presente nei dischi di sistema, solo che legge i file compattati, lasciandone ovviamente inalterato il contenuto originale.

Lo stesso tipo di azione può essere applicato a **file grafici** in formato **Iff** compattati con PowerPacker.

Per visualizzarli senza la necessità di doverli prima scompattare, andrà in questo caso adoperato il programma **PPShow**.

E non è finita.

Esiste anche un PPtype per supplire al comando type, che come questo (in realtà con qualche opzione in più) può stampare su schermo o su carta il contenuto di un file-testo crunchato.

E infine **PPAnim**, che opera sulle animazioni aderenti allo standard Anim (quello di **Dpaint III**, per intenderci) mostrandole a video, anche in questo caso lasciandole compattate.

Grazie all'ausilio di questi tools (ai quali sicuramente se ne affiancheranno presto altri), e alla potenza di PowerPacker, saranno insomma sempre più rari i file uncrunched nei nostri floppy e hard disk, e anche quelli... destinati a scomparire presto.

✔ Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ HELP
❑ Ms - Dos				
✔ Recensioni				
✔ Hardware				
❑ Software				
❑ Applicazioni				
❑ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
❑ Assembly				

Una scheda con prestazioni mozzafiato imprime una definitiva svolta professionale alla grafica di qualunque modello Amiga

E fu subito colore!

< di Domenico Pavone >

< Schede grafiche per Amiga >

La grafica, da sempre, ha costituito l'indiscusso cavallo di battaglia di ogni modello Amiga.

E non certo a torto: prestazioni come quelle disponibili anche nell'uso più comune del nostro amato computer, su altre macchine di ben maggiore diffusione (leggi: Pc) sono soltanto avvicinabili con l'ausilio di schede grafiche di recente produzione.

Per quanto ciò possa fornire un valido appiglio ai sostenitori di fiere battaglie verbali in favore del 16 (oppure 32) bit targato Commodore, occorre però fare qualche considerazione.

Per cominciare, proprio il superamento di certi handicap grafici da parte della concorrenza pone l'accento sui precisi limiti fisici legati all'attuale hardware di Amiga, forse ancora più che sufficienti per reggere il confronto con qualunque altro computer, almeno ad un livello di base, ma finora la caratteristica principale di Amiga era stata proprio quella di consentire, a costi notevolmente inferiori, prestazioni di qualità professionale.

L'evoluzione tecnica, d'altra parte, ha innalzato quella soglia qualitativa che definiva idealmente il confine tra amatoriale e professionale, richiedendo anche ad Amiga quel "qualcosa in più" che possa fare la differenza.

Entrando nel merito, uno dei limiti cui prima si accennava è rappresentato dal **numero di colori visualizzabili** in una stessa immagine grafica.

Per essere più concreti, si pensi al menu introduttivo di **Deluxe Paint IV**, il tool grafico sicuramente più noto a tutti i possessori di Amiga (e recensito in questo stesso numero della rivista). Quel menu, in pratica, schematizza tutte le reali capacità grafiche del computer, consentendo di associare, ad ogni risoluzione di schermo, un certo numero di colori, fino a un massimo di **64** in bassa risoluzione, oppure di **16** in hi-res.

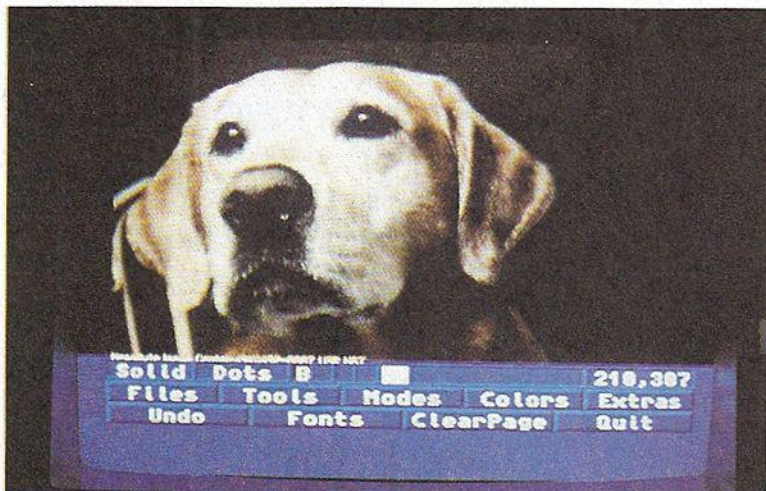
Per qualcosa di più evoluto è sfruttabile il cosiddetto modo **Ham** (Hold and modify), grazie al quale è possibile toccare il limite estremo di **4096** varianti cromatiche.

Che non sono certo poche, ma... avete realmente mai visto una qualche schermata, anche in Ham, che non mostri quel "non so che" di artificioso? Accettabilissimo, per carità, ma in ogni caso lontano da una vera qualità fotografica: quella, per inciso, che sarebbe richiesta in più di un'occasione se si considera un campo di applicazione **professionale**.

Ed eccoci al punto. Per qualcosa del genere, che costituisca una **vera** svolta professionale, il nostro pur valido Amiga non può farcela da solo, necessita di un valido ausilio hardware come quello rappresentato dalla Colorburst, la scheda grafica a 24 bit commercializzata dalla Mast.

Carta d'identità

Lo stesso nome, con una interpretazione sicuramente non azzardata, illustra chiaramente le finalità della scheda. Il termine **burst**, infatti, può essere tradotto dall'inglese con *esplosione*, ed *esplosione di colori* è un'espressione che senza dubbio definisce perfettamente le caratteristiche dell'hardware.

Ma non sarà certo un nome, per quanto azzeccato, a costituire un valido elemento di giudizio, per cui è opportuno porre

Un'immagine, anche se "tradotta" da risoluzioni inferiori, con Colorburst diventa tutta un'altra cosa..

l'accento sulle prestazioni consentite dai già accennati 24 Bit.

Che, per i meno esperti, stanno a significare: **24 bit a diposizione per memorizzare il colore di ogni singolo pixel**. Dato per scontato che si sappia tutti cosa sia un pixel, sarà sufficiente un rapido calcolo per capire di quanti colori si potrà disporre. Basterà elevare 2 alla ventiquattresima potenza, per un risultato da capogiro: **16.777.216**.

Si, avete letto bene, sono proprio quasi **17 milioni**! Niente male come premessa, vero?

Da un punto di vista fisico, Colorburst si presenta come un (quasi) banale scatolotto dall'aspetto molto spartano, caratterizzato da due spie luminose sul frontalino, e due connettori a 23 pin sul posteriore, uno maschio e uno femmina. Questi, senza possibilità di errore, consentono il collegamento all'uscita video del computer tramite un cavetto fornito in dotazione, ed alla normale porta RGB del monitor (sfruttando lo stesso cavo che normalmente va alla porta video del computer). La scheda, inoltre, viene fornita con un suo **alimentatore esterno**.

L'installazione risulta dunque estremamente semplice, e, cosa più importante, **indipendente dal modello Amiga posseduto**: dal "piccolo" 500 fino al super accessoriato Amiga 3000.

Una volta acceso il computer, la scheda farà sentire la sua presenza con un ronzio legato alla ventola di raffreddamento posta al suo interno, e si attiverà da sola, automaticamente, ogni qualvolta un programma che necessita della grafica a 24 bit ne richiederà l'intervento.

Prima di proseguire, è necessario però aprire una breve parentesi. Come si è gia' detto, e del resto non può che essere così se si considera anche il prezzo della scheda (poco meno di 1 milione e mezzo di lire), si sta descrivendo un prodotto realmente professionale, particolarmente adatto per chi intenda utilizzare la grafica di Amiga non certo per esibire la schermata iniziale di un floppy. Il che significa, in altre parole, che è richiesto un supporto hardware di un certo livello.

Vero è che la scheda può essere installata anche in un Amiga 500, ma... non certo in configurazione base.

Un primo requisito riguarda la cosiddetta **memoria Chip**, nei modelli più vecchi (tanto 500 che 2000) limitata a 512 Kb. Per un uso proficuo della Colorburst, **è praticamente indispensabile disporre di 1 Megabyte** (anche se le risoluzioni più basse possono funzionare con soli 512 Kb), direttamente implementati nei **modelli Amiga più recenti** oppure ottenibili **sostituendo il chip Agnus**.

Meno rilevanza (ma sempre meglio averne in abbondanza), assume invece la più tradizionale memoria Fast, in quanto la scheda dispone di 1.5 Mb di RAM video, attraverso la quale gestisce prestazioni di assoluto rilievo, grazie anche alla presenza di un proprio processore (Vlsi a 28 Mhz): in altre parole, è come disporre di un altro computer, altamente specializzato nella manipolazione della grafica a 24 bit.

Un altro aspetto da prendere in considerazione è poi la presenza di memorie di massa adeguate.

Detto più banalmente: senza un hard disk, si limiterebbe notevolmente il campo di azione della scheda, che raggiunge limiti davvero notevoli. Per rendersene

Uno scatolotto dall'apparenza innocua, nasconde prestazioni da capogiro

conto, basterà elencare brevemente le risoluzioni video (in pixel) supportate dalla Colorburst, che tra l'altro si adegua automaticamente allo standard Pal europeo:

320 x 256
384 x 296
320 x 512
384 x 580
640 x 256
768 x 296
640 x 512
768 x 580

Per la cronaca, tutte le risoluzioni da 320 a 384 pixel orizzontali permettono la visualizzazione (e manipolazione, col software che vedremo tra breve) contemporanea di **2** schermi(!).

Tornando al problema di adeguate memoria di massa, qualche altro calcolo ne chiarirà i contorni. Prendiamo in considerazione la più bassa delle risoluzioni possibili, 320 x 256, e proviamo a definire lo spazio su disco che occuperanno i dati della schermata.

Poiché ad ogni pixel corriponde un bit, avremo una base di 320 x 256 = 81920 bit. A questi, poi, dovremo aggiungere i 24 bit per **ognuno** che ne determinano il colore, per cui...

81.920 x 24 = 1.966.080

...pari a...

245760 byte (cioè: **240 Kb**).

Una dimensione ancora accettabile per un floppy disk, anche se la lentezza di caricamento risulterebbe già abbastanza esasperante.

Altro menu di CbPaint, che non riesce a nascondere la meraviglia dei 16 milioni (e rotti) di colori disponibili

Risoluzioni maggiori, però, rendono proibitivo l'uso dei floppy, e proprio per limiti fisici: una schermata di 640 x 512, ad esempio, necessita di ben **960 Kb**!

Limiti fisici a parte, sarebbe comunque un peccato vincolare la Colorburst in limiti angusti come quelli consentiti dai floppy.

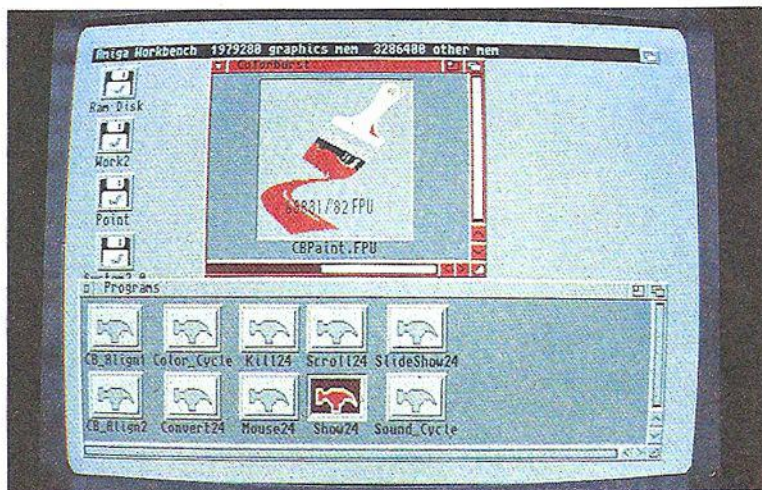
Basterà rimirare una visualizzazione a 24 bit per rendersene conto... e anche innamorarsene: Amiga sembrerà un computer da fantascienza, se si pensa al rapporto prezzo/qualità della scheda.

Inoltre le prestazioni della Colorburst non si limitano alla sola visualizzazione di immagini statiche: la complessa struttura hardware consente anche la gestione di **animazioni** (sempre a 24 bit) che possono richiedere svariati Megabyte di dati, o la realizzazione di effetti particolari come il **color cycling**, la **solarizzazione**, l'**inversione** della palette colore in tempo reale, il tutto senza impegnare la memoria di Amiga, utilizzando le proprie risorse interne.

In certi casi, come per esempio nello scrolling di uno schermo a 24 bit, sostituendosi necessariamente all'hardware del computer, che non sarebbe in grado di gestire con sufficiente velocità l'enorme mole di dati: è tutto dire!

Uno dei menu del programma Cbpaint, che mostra alcuni dei formati grafici importabili

Il software

La scheda Colorburst viene fornita unitamente ad un manuale abbastanza esauriente, e a due floppy: uno contenente 3 immagini compattate con un archiviatore (Lharc), l'altro riservato ad una serie di test e utility, comprendenti tra l'altro un completo programma di *paint* su grafica a 24 bit.

Il software fornito con la Colorburst, visto da ambiente Workbench

E' anche presente un file che consente l'installazione automatica su hard disk, e che provvede anche (se lo si desidera) a scompattare i file grafici e collocarli in una directory di proprio gradimento della periferica. Con la stessa procedura, come descritto dal manuale, è possibile effettuare l'installazione dei programmi su floppy, anche se, come già detto, non è decisamente consigliabile.

Il software, per così dire spicciolo, può essere suddiviso in due categorie: una comprendente le **utility**, e una riservata a testare la scheda e a sincronizzare il suo output con quello del monitor. Operazione, questa, non sempre necessaria, ma che comunque può essere effettuata abbastanza facilmente.

In pratica, basterà lanciare (da Shell o Cli) vari programmi che proporranno sullo schermo diverse gradazioni di grigio o di colore, uno scrolling hardware e un cycling: se l'effetto non è perfettamente visualizzato, occorrerà agire su una vite posta sul retro della scheda fino ad ottenere la resa ottimale.

A onor del vero, abbiamo provato Colorbust su tre diversi computer (con monitor diversi), e in *nessuna* occasione è risultato necessario agire sul trimmer di sincronizzazione.

I vari tools, invece, rendono possibile eseguire su grafica a 24 bit una serie di operazioni che i normali programmi non riuscirebbero ad espletare. Si disporrà quindi di un tradizionale **Show** per visualizzare una singola schermata e di **Slideshow** per sequenze di immagini, e sullo stesso piano può essere considerato anche **Cycle**, per ruotare il colore (sfruttando il processore della Colorburst).

Meno tradizionale invece **Sound Cycle**, che consente di sincronizzare il cycling del colore con un input musicale in forma di Sample (un qualunque campionamento in standard **Perfect Sound**). L'effetto, estremamente simpatico, può grosso modo essere paragonato a quello delle luci psichedeliche, con una pulsazione dei colori davvero degna di nota.

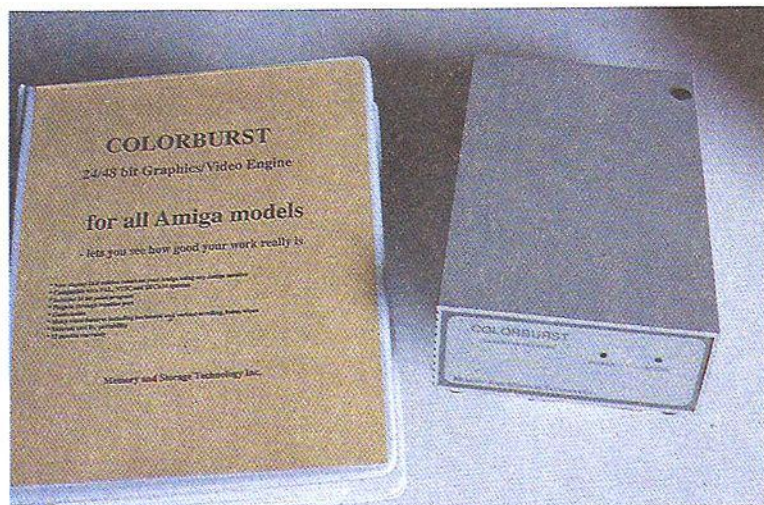
Un altro comodo programma (**Convert**), permette di operare una importante conversione, quella **da** Iff24 **a** Colorburst Fast Format.

Si tratta di due differenti modi di memorizzare i dati riguardanti la grafica: uno standard (l'Iff, che per la grafica a 24 bit viene comunemente definito Iff24), ed uno esclusivo della Colorburst (il **Cff**).

Ma visto che si è giunti a parlare di formati, non resta che accennare al principale programma di manipolazione grafica collegato all'uso della Colorburst.

CbPaint

Pur se di diversa impostazione, questo package può essere paragonato, nelle sue funzioni, agli equivalenti **DPaint**, **Photon Paint** e similari della grafica normale: un insieme di tools per automatizzare, o comunque facilitare, i più comuni interventi di ritocco (o creazione ex novo) su una immagine a 24 bit.

Il programma, attivabile da Workbench, si sviluppa in una serie di menu accessibili via mouse direttamente dalla sezione inferiore dello schermo, che può essere in qualunque momento esclusa

La scheda Colorburst con il suo manuale, all'interno del quale trovano posto anche i due floppy a corredo

dalla visualizzazione premendo il pulsante destro del mouse.

Senza soffermaci su tutte le opzioni disponibili, del resto ampiamente descritte nella manualistica e spesso di intuitiva applicazione, vanno citate in particolare alcune voci del menu **Modes**, che rivestono particolare importanza nel trattamento di questo tipo di grafica.

Avendo a che fare con 16 milioni e rotti di colore, è infatti impensabile gestire (per esempio) la transizione da uno all'altro (il cosiddetto sfumato) pixel per pixel. Allo scopo, quindi, è indispensabile sfruttare tool come **Gradient**, che provvede automaticamente, così come **Trans** creerà un effetto di trasparenza tra il brush attivo e lo sfondo.

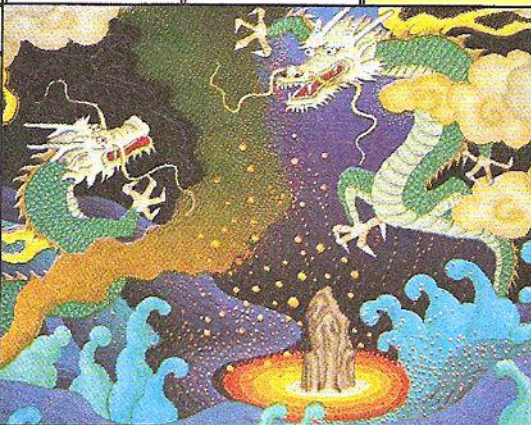
Già, perché sono comunque disponibili quelle peculiarità tipiche di ogni programma di paint come appunto il **Brush**, una sezione di schermo che funge da pennello (naturalmente a 24 bit!), il **Fill**, il tracciamento di **poligoni** e **cerchi**, un menu **Palette** decisamente ricco, la possibiltà di stabilire il cosidetto **Dithering**, uno **Zoom** di tutto rispetto ed altre funzioni.

Tutte opzioni il cui effetto, avendo a che fare con Colorburst, risulterà sminuito da un semplice descrizione: impossibile rendere la perfezione di quanto appare realmente sullo schermo.

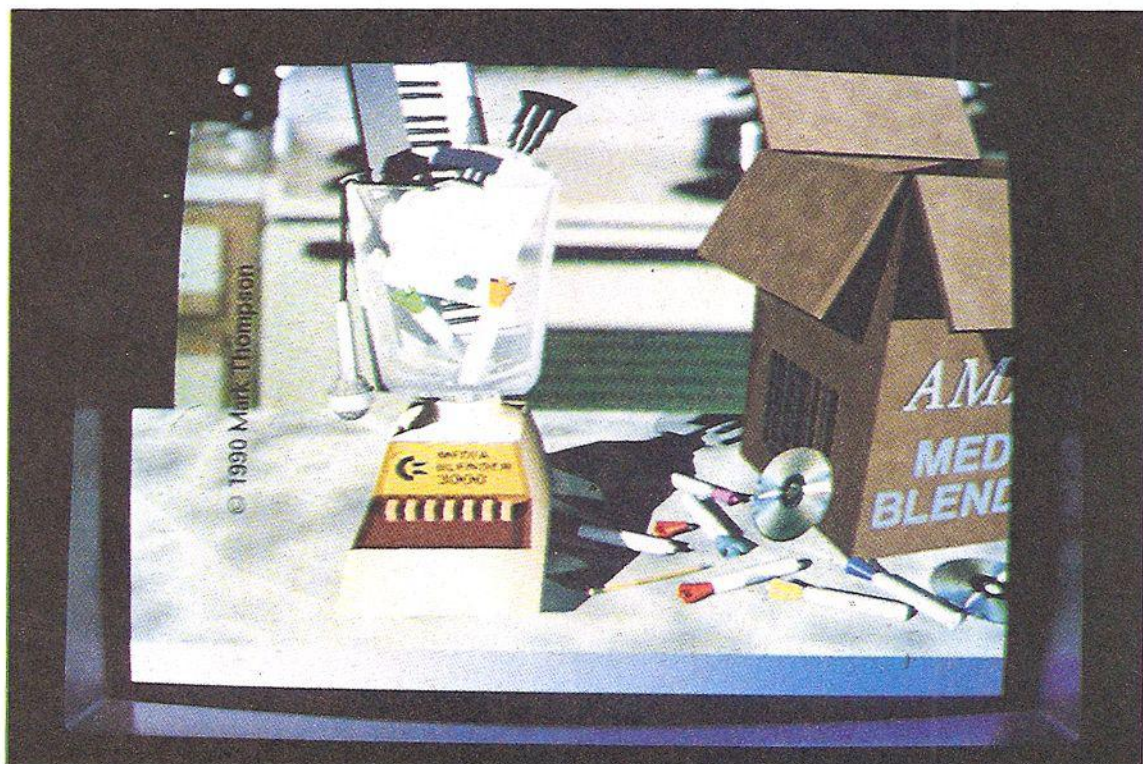
Qualche parola va però spesa a proposito degli standard di formato.

Cbpaint, come già accennato, consente il caricamento e il salvataggio delle immagini in un suo formato compatto e veloce, nei suoi menu definito come **Fast**, ma può risultare utile poterne importare anche altri, non necessariamente a 24 bit.

La scelta possibile in fase di Load, onore al merito del software (comunque in evoluzione: noi abbiamo esaminato la release **1.01**), è ottima: l'Iff normale (da 1 a 5 bitplane) e quello a 24 bit (Iff24), il Tiff tipico dei Macintosh, e ancora **Turbo** (Imagine e Turbo Silver) e **Rend** (Caligari).

Dopo l'elaborazione, l'immagine può poi essere salvata, oltre che ai due consueti modi Fast e Iff24, anche nei formati **Targa** e **IP** (Digiview).

Questi ultimi sono certo più che sufficienti per qualunque ulteriore conversione, se si considera che alcuni comuni programmi Amiga quali Transfer24, Imagelink oppure The Art Departement possono accettare in input uno o più dei suddetti formati.

Si potrebbe ancora parlare a lungo di Colorburst e del software in sua dotazione, ma probabilmente sarebbe un continuo ripetersi di elogi ed espressioni di meraviglia, che solo una visione diretta del monitor di Amiga potrebbe suffragare.

Del resto, basta un aggettivo nella sua più completa accezione per riassumerne le prestazioni: professionale, a tutti gli effetti.

L'impressionante definizione fotografica di una immagine a 24 bit sullo schermo del monitor

✔	Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ HELP
❑	Ms - Dos				
✔	Recensioni				
❑	Hardware				
❑	Software				
❑	Applicazioni				
❑	Programmazione				
	❑ Dos				
	❑ Pascal				
	❑ C				
	❑ Basic				
	❑ Assembly				

Il più noto programma grafico per Amiga è ancora più potente.

Deluxe Paint IV

⇨ < di Luigi Callegari >

< Abbiamo parlato di pacchetti grafici anche su C. C. n. 74, n. 83, n. 86 >

Deluxe Paint della Electronic Arts è da sempre considerato lo standard *de facto* nel mondo Amiga, essendo utilizzato da artisti e professionisti più di qualunque altro pacchetto grafico.

Sino a **DPaint III** (recensito sul lontano Computer Club n. 74), la carenza maggiore di questo rivoluzionario programma grafico era certamente la mancanza del modo **HAM**, che costringeva a rivolgersi a programmi come Digipaint III o Photon Paint (ora **Spectra Color**) per sfruttare le 4096 tinte di Amiga contemporaneamente.

Oltretutto, questi pacchetti hanno il difetto di funzionare SOLO in modo HAM, rendendo difficile, se non impossibile, creare agevolmente e tranquillamente schermi a soli 32 colori, o di lavorare in alta risoluzione, e costringendo grafici ed artisti ad un uso combinato col tradizionale Deluxe Paint e/o altri tool grafici di conversione.

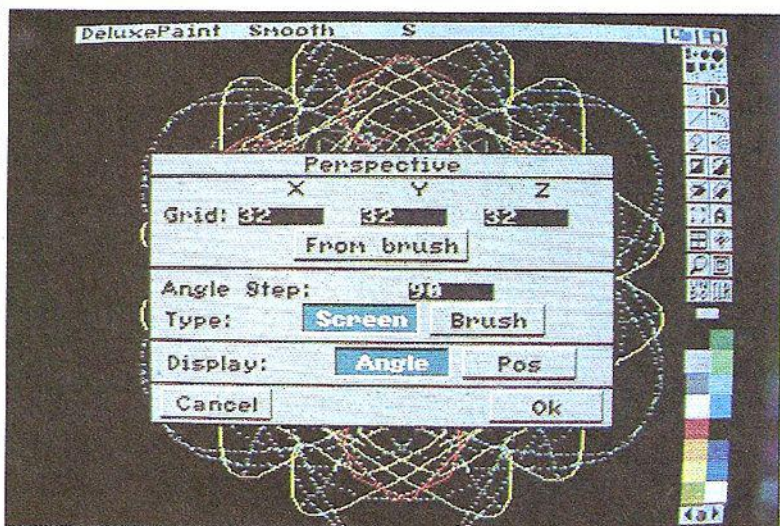
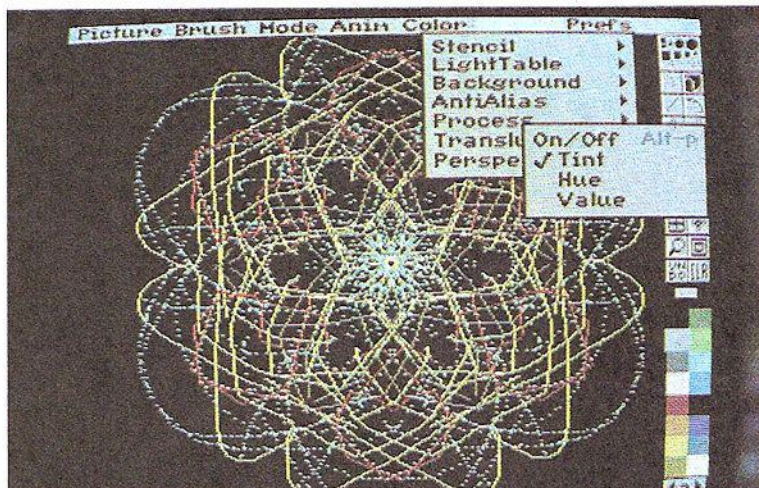
La recentissima versione di Deluxe Paint IV, la cui distribuzione è avvenuta ufficialmente in Europa negli ultimi giorni di settembre, fornisce invece non soltanto il modo HAM in 4096 colori, con un completo supporto per le **animazioni grafiche**, ma anche tutti i tradizionali modi schermo, compresi il **superbitmap** e l'**overscan**.

Inutile dire che il pacchetto è stato completamente provato per funzionare perfettamente sotto Kickstart e Workbench **V 2.0** e con i nuovi chip custom montati di serie in **Amiga 3000** (e, si spera, in un prossimo futuro su tutti gli Amiga).

❑❐

Disegno

I ritocchi, rispetto a Deluxe Paint III, a prima vista appaiono piuttosto marginali, se ci si limita ad osservare lo schermo di lavoro del programma, ma effettivamente sono numerosi.

Innanzitutto il menu di tavolozza prevede nuovi modi per generare e mescolare colori: componenti RGB, Hue Saturation e

Value come sempre, ma anche la possibilità di tracciare i colori **direttamente** sopra una specie di tavolozza di mistura per creare nuove tinte ed ombre ed utilizzarle subito, esattamente come farebbe un pittore, unite però alle sofisticate funzioni elettroniche di Amiga.

I *range* di colore ora possono essere stabiliti più facilmente: non più colori iniziali e finali sulla tavolozza, ma da qualunque punto dello schermo, lasciando a DPaint il compito di calcolare il numero richiesto di tinte intermedie per creare la gamma secondo il modo video attuale.

Oltretutto, la finestrella di controllo della tavolozza ora può essere lasciata attiva a video mentre si continua a disegnare sullo schermo, consentendo di verificare visivamente quanto si sta facendo prima di fissare le nuove regolazioni cromatiche.

I gradienti di colori possono ora andare a riempimento orizzontale, verticale e si possono creare linee di colore, gradienti di contorno, sagome di riempimento o di evidenza circolari e concentriche.

Particolarmente notevole la nuova funzione di menu **Process**, che consente una elaborazione dei colori vagamente simile a PIXmate od a DigiPaint III: si possono colorare aree definite con un nuovo range di colori, modificare la componente Hue di colori a video per rendere un particolare del disegno più chiaro o più scuro agendo sui colori invece di ridisegnarlo o ricolorarlo a mano. E' anche previsto l'effetto **Translucency**, che consente di lasciare brush sul disegno in modo che risultino parzialmente trasparenti.

Tra gli altri effetti e funzioni, comunque troppo numerosi e complessi per essere descritti in modo esauriente in queste pagine, troviamo l'**anti aliasing** automatico per evitare la segmentazione delle linee e il mescolamento automatico dei colori durante il disegno.

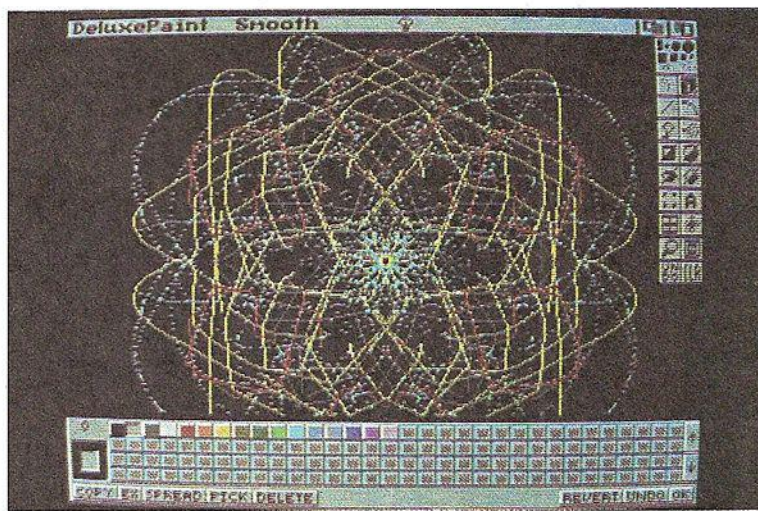
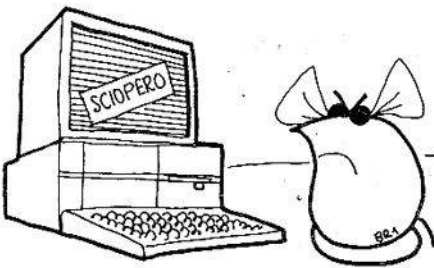
In effetti, in un Amiga normale molte di queste operazioni complesse, in particolare nel modo HAM a 4096 colori, richiedono parecchio tempo per essere eseguite; pertanto diventa importante per chi usa Amiga professionalmente o, comunque, ad un livello evoluto, dotarlo di una scheda acceleratrice (a meno che si abbia già un Amiga 3000, ovviamente!).

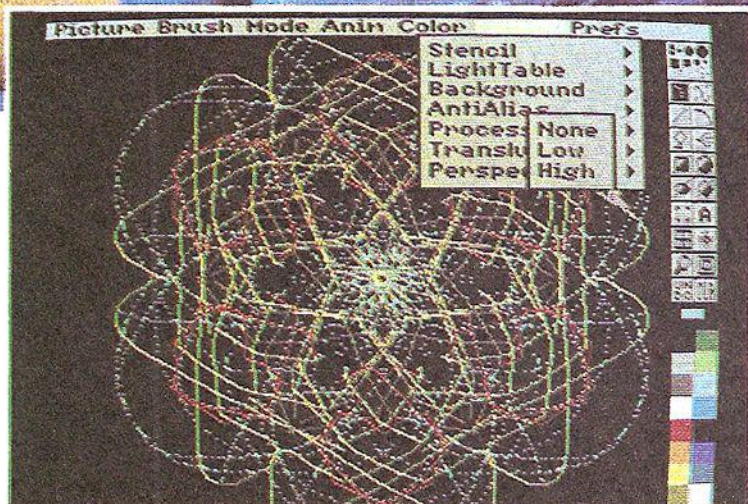
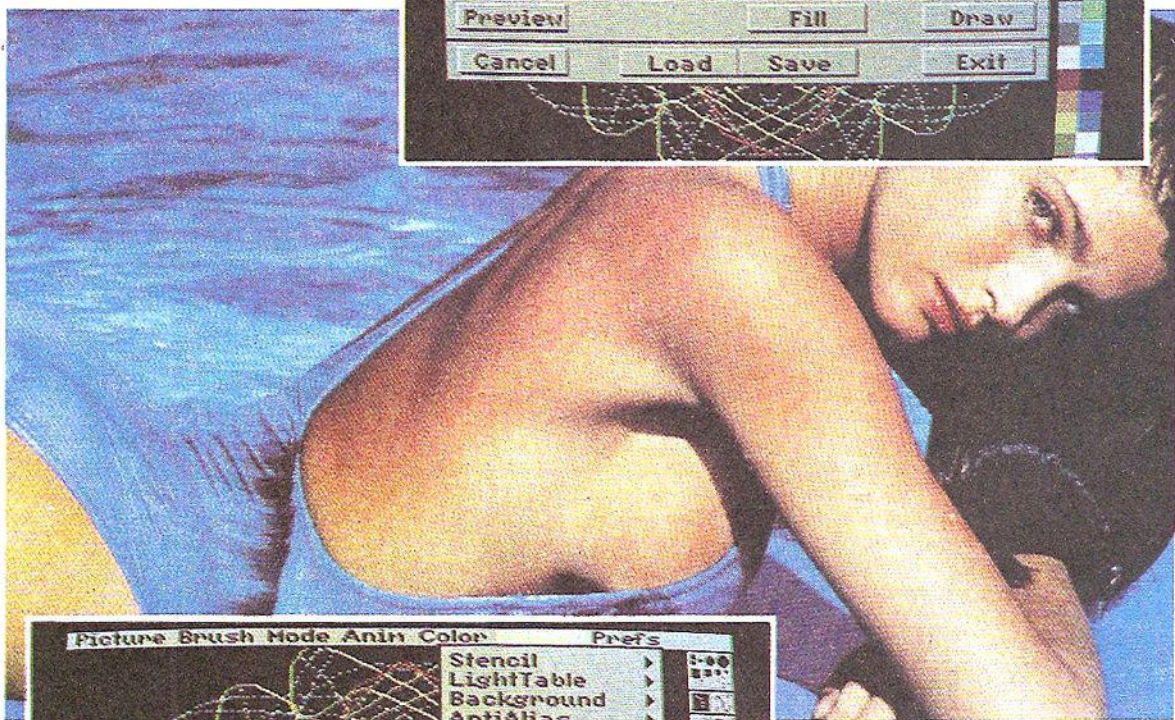
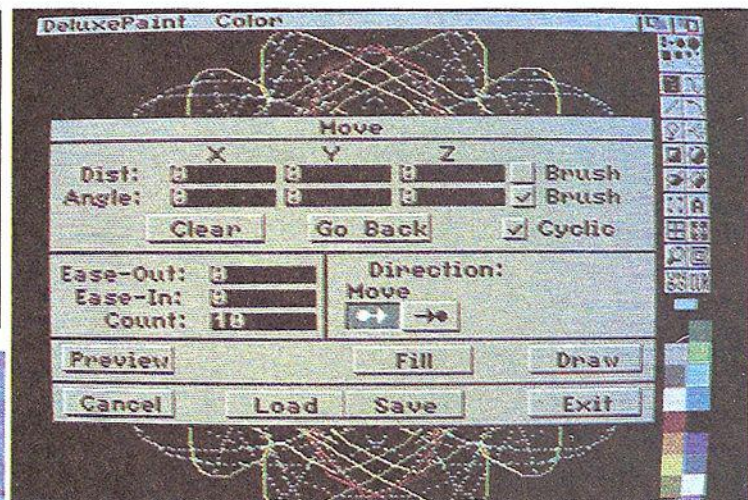
Animazione

Il nuovo modo di animazione di Deluxe Paint IV si chiama **Animpainting** e consiste in un sistema di scambio automatico di fotogrammi di una animazione durante il disegno, evoluto da quanto già presente in DPaint III.

Ora però è possibile creare, ad esempio, animazioni che procedono lungo una linea semplicemente prelevando la brush, selezionando una linea punteggiata dalle opzioni di tracciatura delle rette e tenendo premuto **Alt** mentre, con il mouse, si tira la linea: la brush animata seguirà la linea stessa, dal momento che il programma ciclerà i fotogrammi inserendo ciascuna immagine nello schermo corrispondente.

Il controllo delle animazioni è ora semplificato ed assomiglia di più a programmi come **Movie Setter** (recensito in CCC n. 81) che al rudimentale sistema di Dpaint III, essendo dotato di un vero e proprio **pannello di controllo** con funzioni di redazione dell'esecuzione dei fotogrammi, nonchè un tavola di controllo della luce sull'animazione.

In questo modo si possono vedere i due o tre fotogrammi precedenti oppure

la pagina vuota posta sotto la schermata attuale.

Sebbene parecchie funzioni siano identiche a quelle già presenti in Dpaint III, a parte però una intera nuova gamma di comandi per lo spostamento sui fotogrammi dell'animazione, esse ora risultano più facilmente utilizzabili e quindi più potenti.

Conclusioni

L'equazione è semplice. Se Deluxe Paint III era un programma eccellente e molto apprezzato, tenendo conto che la nuova versione ha soltanto aggiunto nuove funzioni e caratteristiche o perfezionato e riordinato quelle già presenti, inserendo addirittura il tanto agognato modo HAM con un completo supporto all'animazione, si può dedurre che DigiPaint IV avrà un enorme successo e sarà usato come strumento creativo di lavoro professionale da tutti i possessori di Amiga, artisti e non.

✔ Amiga	❑ Principianti	✔ Esperti	❑ Tutti	✔ Help
❑ Ms - Dos				
❑ Recensioni				
❑ Hardware				
✔ Software				
❑ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
✔ C				
✔ Basic				
✔ Assembly				

Che cosa contengono le librerie di sistema di Amiga? Vediamo di scoprirlo insieme

I misteriosi file .Bmap di Amiga

< di Pasquale Paparella>

< Come creare ed interpretare correttamente i file .bmap >

Le librerie di sistema sono un insieme di routine, molte delle quali scritte in linguaggio macchina, raggruppate secondo l'uso cui sono destinate.

Indipendentemente dal linguaggio di programmazione, per usare una qualsiasi di queste routine bisogna *aprire* la libreria cui appartiene, *rintracciarla* tramite il valore di **OFFSET**, *passare* ai giusti registri hardware i relativi parametri e, infine, *richiudere* la libreria quando non serve più.

Il valore di OFFSET (sempre **negativo**) indica di quanto spostarsi rispetto al puntatore alla libreria in questione affinché venga rintracciata la routine voluta.

AmigaBasic permette di accedere facilmente alle librerie rispettando la procedura appena descritta; il listato più breve di queste pagine ne è un conferma.

Fa uso del comando **WindowLimits (Window, minwidth, minheight, maxwidth, maxheight)** della "**intuition.library**" e si limita semplicemente a far variare il dimensionamento della finestra entro i limiti minimi e massimi impostati.

Si può notare l'uso di LIBRARY "intuition.library" per aprire la libreria e di **LIBRARY CLOSE** per chiuderla mentre *Window, minwidth, minheight, maxwidth, maxheight* sono i parametri che vengono passati ai registri hardware; ma del valore di OFFSET, o dei registri stessi, nel listato nemmeno l'ombra.

Questi parametri sono contenuti, infatti, nei file **.bmap**.

Quando si chiede ad AmigaBasic di aprire una libreria, si chiede in pratica all'interprete di controllare se esiste il corrispondente file .bmap; se non lo trova, emette un segnale d'errore.

I file .bmap non sono in dotazione con l'interprete, ma bisogna crearli, operazione questa relativamente facile.

Sul dischetto **Extras** nella directory **FD1.3** sono presenti tutti i file necessari, mentre nella directory **BasicDemos** (sempre del dischetto Extras) troviamo il programma **ConvertFD**.

Lanciato con un doppio click sulla sua icona ConvertFD, quest'ultimo chiede il nome del file **.fd** da convertire in file **.bmap** e, subito dopo, il **nome** da assegnare al file .bmap prodotto.

Supponiamo, per esempio, che necessita il file **Timer.bmap**; se il file **timer_lib.fd** è contenuto nella directory FD1.3 del dischetto Extras e se vogliamo il file nella **Ram:** con il nome finale di **Timer.bmap**, dobbiamo rispondere con...

```
Extras:fd1.3/timer_lib.fd
```

...alla prima domanda e con...

```
ram:timer.bmap
```

...alla seconda.

Subito dopo avremo il file cercato nel path voluto.

```
78 4F 70 65 6E      00 FF E2 02 03      00
x  O  p  e  n          -30   d1 d2

78 43 6C 6F 73 65   00 FF DC 02         00
x  C  l  o  s  e       -36   d1
```

Ecco i primi due valori visualizzati elaborando il file Dos.bmap presente nella directory Libs del disco Extras.

L'uso dei file .bmap può generare qualche problema all'interprete su **dove** trovarli, specialmente se si usa un solo drive.

Un'istruzione tipo `LIBRARY "dos.library"`, infatti, non creerà problemi **solo se** il relativo file .bmap si trova nella stessa directory del programma oppure nella directory **Libs** del dischetto che ha avviato il sistema.

Se, invece, il file .bmap è memorizzato, per esempio, su di un dischetto di nome **miodisco** nella directory **libreria**, l'istruzione precedente diventerà...

```
LIBRARY "miodisco:libreria/dos.bmap"
```

A pagina **261** del manuale AmigaBasic si può leggere che il formato di ogni routine nel file .bmap è composto dal **nome** della routine stessa, seguita da un **byte nullo**, dal valore di OFFSET, dai **registri hardware** cui passare i parametri ed, infine, da un altro **byte nullo**.

Il manuale, inoltre, commenta il valore assegnato ad ogni registro. L'unico modo per sbirciare all'interno dei file .bmap è quello di servirsi del comando **Type** con opzione **hex** (oppure, naturalmente, di un Disk Editor).

Se, per esempio, digitiamo da Shell...
`type intuition.bmap hex`
e andiamo alla ricerca di **WindowLimits** usata nel listato 1, troveremo i seguenti valori in formato esadecimale...

```
59, 69, 6E, 64, 6F, 77, 4C, 69, 6D, 69, 74,
73, 00, FE, C2, 09, 01, 02, 03, 04, 00
```

...mentre nella parte destra, riservata alla traduzione in ASCII di ciò che è possibile tradurre, troveremo il nome WindowLimits seguito da alcuni puntini, e da lettere apparentemente senza senso, prima di trovare il nome della routine successiva.

```
LIBRARY "intuition.library"
WINDOW 2,"",(20,20)-(250,100),17
PRINT "Modifica le dimensioni"
PRINT "Premi poi un tasto"
PRINT "per finire"
a&=WINDOW(7)
CALL WindowLimits&(a&,100,50,530,150)
w$="":WHILE w$="":w$=INKEY$:WEND
WINDOW CLOSE 2
LIBRARY CLOSE
END
```

Il brevissimo programma dimostrativo, in AmigaBasic, sull'ultilità dei file .bmap. Prima di dare Run è però indispensabile esaminare l'articolo di queste pagine.

Sono proprio quei valori, incomprensibili per type (ed anche per le menti normali...), che contengono il valore di OFFSET e i registri hardware.

Il secondo programma

Il listato "**.bmap_decodificatore**" è in grado di ricavare, per ogni routine presente in un file .bmap, tutto ciò che in esso è contenuto: nome, offset e registri hardware.

Fa uso di 4 routines della **dos.library**, tre delle quali, essendo funzioni, vanno dichiarate all'inizio del programma.

Apre poi la dos.library e, tramite la funzione **Execute**, copia in **Ram:** il comando **Type** per evitare di accedere più volte alla directory **C** nel caso in cui si intenda esaminare più di un file .bmap che si trovasse su un altro dischetto.

Viene quindi richiesto il nome del file .bmap da esaminare e, tramite, il comando Type con opzione **hex**, si crea in Ram: un file dal nome **libreria**.

Tramite la funzione **IoErr** si controlla se il nome fornito è esatto.

Nel caso contrario, cioè se non si riesce a trovare il file, il programma viene deviato alla subroutine di nome **sbaglio**, e da qui ricomincia.

Se non ci sono stati errori, e se in Ram: è presente il file "libreria" prodotto dal comando type, viene chiusa la dos.libreria, aperto normalmente il file ram:libreria e, per ogni LINE INPUT, vengono prelevati solo i caratteri esadacimali riuniti in un'unica variabile. Quando NOT EOF(1) rileva la fine del file ram:libreria, il file stesso viene chiuso ed inizia la decodifica dei valori esadecimali contenuti nella variabile secondo lo schema:
`nome, 00, offset (4 bit), eventuali registri, 00`

A prima vista l'utilità pratica di un simile programma sembrerebbe piuttosto scarsa.

Tuttavia, a livello didattico, è utilissima sia per comprendere come sono strutturati i file .bmap, sia (soprattutto) per ricordare in che modo realizzarli, partendo dal dischetto Extras.

Coloro che muovono i primi passi nel favoloso Assembler del 68000 hanno certamente bisogno di studiare i valori di offset. Non possedendo i vari "ROM Kernel Manual" (estremamente cari...) l'unico modo per procurarsi tali valori è quello che la stessa Commodore offre gratuitamente: i file.bmap. A patto, però, di saperli comprendere.

Il listato in AmigaBasic "Decoder .bmap". Tener presente che le righe sottolineate, apparentemente disposte una sull'altra, devono esser invece digitate su un'unica linea. Il programma, che avrebbe dovuto esser riportato su una sola colonna, è stato stampato su due colonne per evitare spreco di spazio.

```
'***********************************
' OFFSET per AmigaBasic
' di  Pasquale PAPARELLA
' 1991
'***********************************

'DICHIARA LE FUNZIONI E APRE LA LIBRERIA DOS

DECLARE FUNCTION xOpen& LIBRARY
DECLARE FUNCTION Execute% LIBRARY
DECLARE FUNCTION IoErr& LIBRARY
```

```
LIBRARY "dos.library"

'COPIA IN RAM TYPE

copia$="copy c/type to ram:"+CHR$(0)
y$="nil:"+CHR$(0)
out&=xOpen&(SADD(y$),1006)
e%=Execute%(SADD(copia$),0,out&)
CALL xClose(out&)

'CHIEDE LA LIBRERIA DA ESAMINARE E CREA IN RAM:
'TRAMITE IL COMANDO TYPE IL FILE libreria

inizio:
LOCATE 8,16
PRINT "Digita il nome del file .bmap da
esaminare"
PRINT TAB(21)"( Completo di path di ricerca )"
LOCATE 11,17
LINE (100,88)-(500,88)
LINE INPUT afile$
comando$="ram:type "+afile$+" hex TO
ram:libreria"+CHR$(0)
y$="nil:"+CHR$(0)
out&=xOpen&(SADD(y$),1006)
e%=Execute%(SADD(comando$),0,out&)
CALL xClose(out&)
errore&=IoErr& 'Controlla un eventuale errore di
percorso
IF errore&=205 THEN sbaglio

'APRE IL FILE libreria E PRENDE I SOLI VALORI
ESADECIMALE
'PONENDOLI IN UN'UNICA VARIABILE DI NOME file$

file$="":a$=""
OPEN "I",#1,"ram:libreria"
   WHILE NOT EOF(1)
   LINE INPUT #1,a$
   variabile$=MID$(a$,7,36)
   file$=file$+variabile$
   WEND
CLOSE #1
   CLS

'TROVA IL NOME IN ESADECIMALE

x=1:y=1:t=0
GOTO azzera
loop:
w$=""
WHILE w$<>"00"
w$=MID$(file$,y,2)
IF w$="" THEN esci
IF w$="00" THEN
nomeesadecimale$=nomeesadecimale$ ELSE
nomeesadecimale$=nomeesadecimale$+w$
x=x+1
IF x=5 THEN
y=y+3:x=1
ELSE
y=y+2
END IF:WEND

'ASSOCIA IL PRIMO CARATTERE NULLO
primonullo$="00"

'RICAVA IL VALORE DELL'OFFSET IN ESADECIMALE

FOR z=1 TO 2
w$=MID$(file$,y,2)
offsetesadecimale$=offsetesadecimale$+w$
x=x+1
IF x=5 THEN
y=y+3:x=1
ELSE
y=y+2
END IF
NEXT

'RICAVA I REGISTRI A CUI PASSARE I VALORI

WHILE w$<>"00"
w$=MID$(file$,y,2)
IF w$="00" THEN registri$=registri$ ELSE
registri$=registri$+w$+" "
x=x+1
IF x=5 THEN
y=y+3:x=1
ELSE
y=y+2
END IF:WEND

'ASSOCIA IL SECONDO VALORE NULLO
secondonullo$="00"

'RICAVA IL NOME IN ASCII

k1=LEN(nomeesadecimale$):yy=1:k=k1/2
FOR z=1 TO k
w$=MID$(nomeesadecimale$,yy,2)
GOSUB traduzione
cambio$=CHR$(cambio)
nomeascii$=nomeascii$+cambio$
yy=yy+2
NEXT

'RICAVA IL VALORE DI OFFSET

yy=2
FOR z=1 TO 3
w$(z)=MID$(offsetesadecimale$,yy,1):yy=yy+1
NEXT
zw$=w$(1):GOSUB traduzione1:cambio1=ww
zw$=w$(2):GOSUB traduzione1:cambio2=ww
zw$=w$(3):GOSUB traduzione1:cambio3=ww
offsetdecimale=(cambio1*16*16+cambio2*16+cambio3)-
4095-1

'RICAVA I REGISTRI A CUI PASSARE I PARAMETRI

numeroregistri=LEN(registri$)/3 :yy=2
FOR z=1 TO numeroregistri
registro$=MID$(registri$,yy,1)
'GOSUB trovaregistro
zw$=registro$:GOSUB traduzione1
www=ww :GOSUB conversione
```

```
registrohardware$=registrohardware$+registrohard$
yy=yy+3
NEXT

'STAMPA A VIDEO I VALORI OTTENUTI

IF t=6 THEN GOSUB attesa
COLOR 1,2
PRINT
nomeesadecimale$;TAB(40)primonullo$;TAB(43)offsete
sadecimale$+"  ";registri$;TAB(75)secondonullo$
COLOR 1,0
PRINT nomeascii$
;TAB(43)offsetdecimale;TAB(49)registrohardware$
PRINT
t=t+1

'    AZZERA TUTTI I VALORI

azzera:
offsetesadecimale$=""
registri$=""
primonullo$=""
secondonullo$=""
nomeesadecimale$=""
nomeascii$=""
registrohardware$=""
GOTO loop

'RICAVA I REGISTRI HARDWARE

conversione:
ON www GOTO
hard1,hard2,hard3,hard4,hard5,hard6,hard7,hard8,
hard9,hard10,hard11,hard12,hard13

hard1:
  registrohard$="d0 ":RETURN
hard2:
  registrohard$="d1 ":RETURN
hard3:
  registrohard$="d2 ":RETURN
hard4:
  registrohard$="d3 ":RETURN
hard5:
  registrohard$="d4 ":RETURN
hard6:
  registrohard$="d5 ":RETURN
hard7:
  registrohard$="d6 ":RETURN
hard8:
  registrohard$="d7 ":RETURN
hard9:
  registrohard$="a0 ":RETURN
hard10:
  registrohard$="a1 ":RETURN
hard11:
  registrohard$="a2 ":RETURN
hard12:
  registrohard$="a3 ":RETURN
hard13:
  registrohard$="a4 ":RETURN
'DIVIDE IN DUE IL VALORE ESADECIMALE
traduzione:
  ww$=MID$(w$,1,1):www$=MID$(w$,2,1)
  zw$=ww$ :GOSUB traduzione1
  decimale1=ww
  zw$=www$ :GOSUB traduzione1
  decimale2=ww
  cambio=decimale1*16+decimale2
  RETURN

'DA ESADECIMALE A DECIMALE

traduzione1:
  IF zw$<="9" THEN ww=VAL(zw$)
  IF zw$="A" THEN ww=10
  IF zw$="B" THEN ww=11
  IF zw$="C" THEN ww=12
  IF zw$="D" THEN ww=13
  IF zw$="E" THEN ww=14
  IF zw$="F" THEN ww=15
  RETURN

'PER ESAMINARE UN ALTRO FILE O TERMINARE
esci:
  KILL "ram:libreria"
  COLOR 2,3
    PRINT  "Vuoi esaminare un altro file .bmap ?
(s/n)";
    esci1:
    risposta$=INKEY$
    IF risposta$<>"s" AND risposta$<>"n" THEN
esci1
    COLOR 1,0:CLS
    IF risposta$="s" THEN inizio
  KILL "ram:type"
  END

'ASPETTA LA PRESSIONE DI UN TASTO

attesa:
COLOR 2,3
  PRINT "Premi un tasto per continuare"
  aspetto$="":WHILE
aspetto$="":aspetto$=INKEY$:WEND
  t=0:COLOR 1,0:CLS
  RETURN

'ERRORE NEL PERCORSO DEL FILE .BMAP

sbaglio:
  CLS:COLOR 2,3
  PRINT TAB(20) "Codice errore 205....non trovo
";afile$
  COLOR 1,0:PRINT :PRINT "Premi un tasto
qualsiasi"
  tasto$="":WHILE
tasto$="":tasto$=INKEY$:WEND:CLS
  GOTO inizio

END
```

✔ Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ Help
❑ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
✔ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
✔ Basic				
❑ Assembly				

Il programma inviato da un nostro lettore può essere considerato come una base da cui partire per realizzare, magari, un word processor.

Come ti tratto un testo ASCII

< di Francesco Varone >

< Esaminare C. C. n. 83 > < Si lancia anche una nuova sfida ai lettori! >

Il programma qui pubblicato (per **AmigaBasic**, ma facilmente adattabile ad altri computer e/o linguaggi) è molto semplificato ma decisamente efficiente.

Risponde in pieno alla **sfida** lanciata sul n. **83** in cui si chiedeva un listato che fosse in grado di esaminare un testo memorizzato su disco in semplice formato ASCII e, quindi, di manipolarlo a più non posso (dalla suddivisione in sillabe all'allineamento secondo diverse impostazioni).

Appena lanciato, richiede i dati di cui ha bisogno, cioè il nome del file da stampare (comprensivo di eventuale **path**), la periferica di *output* (**video** o **stampante**), il tipo di allineamento (a **destra**, a **sinistra**, **centrato** tra i margini o **giustificato** sia a destra che a sinistra) e il numero di **colonne** di testo da stampare in ogni pagina (attenzione: *non* numero di caratteri per riga!).

Per il resto pensa a tutto il programma e l'utente deve soltanto premere un tasto per proseguire dopo aver eventualmente cambiato foglio di carta se ha selezionato la stampante.

Optionals

Tra gli input manca la scelta delle dimensioni della pagina, cioè una delle condizioni della sfida, ma l'autore ha preferito tralasciarla per evitare di allungare troppo il programma.

Comunque dovrebbe essere abbastanza facile aggiungere la richiesta della selezione dei margini e, magari, del passo di stampa (pica, elite, espanso, compresso, ma solo per l'output su stampante).

Riguardo alle dimensioni della pagina il programma si regola a seconda della periferica di output che si vuole utilizzare: le variabili **car** e **rig**, che indicano i caratteri per riga e le righe per pagina, vengono impostate a valori appropriati per lo schermo o per la stampante, dopo aver scelto la modalità di output.

Nella riga successiva alla label **L2**, la prima coppia (car = **76**: rig = **20**) indica le dimensioni della finestra di output sul video; la seconda (car = **80**: rig = **60**) quelle di un foglio della stampante.

Analogamente si può modificare la quantità di **spazi** inseriti tra le eventuali colonne (variabile **sp**, proprio sulla stessa riga). Inoltre, se non si dispone della tastiera italiana, si può evitare di battere la **linea 350**, ma consiglio la consulenza di un amico dotato di un'Amiga superiore!

❑ ❑

Come funziona

Ricevute tutte le informazioni, ed aperti i canali di input ed di output, il programma legge (label **Leggi**) un paragrafo alla volta e lo divide in righe di lunghezza massima stabilita, eventualmente giustificate, e le memorizza nel vettore **st$()**.

Il programma elabora in modo differente il paragrafo a seconda di alcune condizioni:

➪ se il file è *terminato*, stampa la parte rimanente del testo e poi termina;

➪ se la riga è *vuota*, la considera pronta per essere stampata e la memorizza così com'è;

➪ idem se il paragrafo è *più corto* del valore massimo consentito (stabilito in **ln**);

➪ se il paragrafo è *più lungo* di una riga e il primo carattere che non entra nel rigo (`MID$ (parag$, ln + 1, 1)`, dove **ln è** la lunghezza del rigo) è uno spazio, prende la prima parte lunga quanto una riga e la conserva per la stampa, mentre rielabora la parte rimanente ignorando, naturalmente, lo spazio (questo punto è un'ottimizzazione in quanto la parte successiva del programma sarebbe stata in grado di fare lo stesso, ma perdendo più

```
REM * ASCII-Processor (AmigaBasic) ***
REM ***     by Francesco Varone      ***
REM ***      1991 by COMPUCLUB       ***


DIM st$(190)
INPUT"Nome File";nf$
OPEN nf$ FOR INPUT AS 1

L1:INPUT"[V]ideo o [S]tampante";out$
out$=UCASE$(out$):sp=4:car=76:rig=20
IF out$="S" THEN car=80:rig=60
IF out$<>"V" AND out$<>"S" THEN L1
PRINT"Allineamento:"

L2:INPUT"[D]estro, [S]inistro, [C]entrale, [B]ilaterale";Al$
Al$=UCASE$(Al$)
IF Al$<>"D" AND Al$<>"S" AND Al$<>"C" AND Al$<>"B" THEN L2

L3:INPUT"Quante colonne";col
IF col<1 OR col>5 THEN L3
ln=INT(car/col):IF col>1 THEN ln=ln-sp
a$="SCRN:":IF out$="S" THEN a$="LPT1:"
OPEN a$ FOR OUTPUT AS 2
WIDTH a$,car

Leggi:
IF EOF(1) THEN CLOSE 1:GOSUB Stampa:END
LINE INPUT#1,ra$:IF ra$="" THEN x$=ra$:GOTO Fatto

Accapo:
IF ra$="" THEN Leggi
IF LEN(ra$)<=ln THEN x$=ra$:ra$="":IF Al$="B"THEN Fatto:ELSE Giusti
s=ln+1
IF MID$(ra$,s,1)=" " THEN x$=LEFT$(ra$,ln):GOTO Dop
FOR a=s+1 TO LEN(ra$):IF MID$(ra$,a,1)=" " THEN max
NEXT

max:s=a

Ripro: GOSUB Sillaba
IF s=0 THEN x$=LEFT$(ra$,ln):ra$=MID$(ra$,ln+1):GOTO Giusti
a$=MID$(ra$,s,1)
IF a$="-" OR a$="=" THEN
  x$=LEFT$(ra$,s):IF s>ln THEN Ripro
  GOTO Dop
END IF
IF a$=" " THEN
  x$=LEFT$(ra$,s-1):IF s>ln THEN Ripro
  GOTO Dop
END IF
x$=LEFT$(ra$,s)+"-"
IF s>=ln THEN Ripro

Dop:
ra$=MID$(ra$,s+1)

Giusti:
IF Al$="D" THEN Fatto
IF Al$="S" THEN x$=RIGHT$(SPACE$(ln)+x$,ln):GOTO Fatto
IF Al$="C" THEN x$=SPACE$((ln-LEN(x$))/2)+x$:GOTO Fatto
WHILE LEN(x$)<ln:fl=0
  IF INSTR(x$," ")=0 THEN Fatto
```

tempo, poiché avrebbe messo in atto la routine di sillabazione più volte);

⇨ in tutti gli altri casi, viene individuato il *primo* carattere di spazio che non rientra nel rigo e, a partire da questo punto della stringa del paragrafo e andando verso l'inizio, vengono ricercati (grazie alla subroutine **Sillaba**) i punti in cui la stringa può essere spezzata affinché venga trovata la lunghezza che rientra nel rigo (ciclo **Ripro**).

Come accennato è la subroutine *Sillaba* che riconosce i punti in cui una stringa di testo può essere "staccata" e cioè:

⇨ tra una sillaba e l'altra (ed è proprio questa facoltà che dà il nome alla subroutine; è il programma principale che inserisce il trattino);

⇨ dove c'è uno *spazio* (tralasciando lo spazio stesso, ma evitando il trattino);

⇨ dove c'è un *trattino* o un segno uguale, lasciando questi caratteri alla fine del rigo e i successivi all'inizio del successivo (si potrebbero includere anche altri caratteri come, ad esempio, vari segni di punteggiatura).

Sillabando

La routine di sillabazione **non** è la stessa del numero **83** di **CC**, in cui veniva proposta la sfida, in quanto serviva un procedimento a ritroso (ritenuto più valido) che elaborasse, cioè, la parola dalla *fine* verso l'*inizio*.

In realtà questa routine elabora una stringa che può contenere anche spazi ed altri caratteri e, appunto, tiene in considerazione questo fatto.

Non è eccezionale, nè completa (ad esempio, non separa gli **iati**) ma, almeno, possiamo definirla infallibile.

La versione implementata nel programma di queste pagine non ha mai separato in modo errato una parola, anche se non è mai riuscita a separare la parola **gioia**.

Per quanto riguarda la giustificazione (label **Giusti**), non presenta un algoritmo molto complesso: inserisce uno spazio alla volta tra ogni parola e la successiva fino a che non raggiunge la lunghezza stabilita in **ln**.

Provvedimenti ancora più semplici vengono presi nel caso si decida un allineamento a destra, sinistra o centrale.

```
  FOR a=1 TO LEN(x$)
    IF MID$(x$,a,1)=" " THEN
      IF fl=0 THEN Ecco
      x$=LEFT$(x$,a)+" "+MID$(x$,a+1):a=a+1
      IF LEN(x$)=ln THEN GOTO Ok
    END IF
    fl=1
  Ecco:
  NEXT
Ok:
WEND
Fatto:x$=LEFT$(x$+SPACE$(ln+sp),ln-sp*(col>1))
st$(ct)=x$:ct=ct+1:PRINT CHR$(12);ct
IF ct=rig*col THEN GOSUB Stampa
GOTO Accapo

Stampa:
rg=INT(ct/col):IF ct/col<>rg THEN rg=INT(ct/col+1)
FOR a=1 TO rg
  FOR b=0 TO col-1
    c=a+b*rg-1
 IF c<ct THEN PRINT#2,st$(a+b*rg-1);:ELSE PRINT#2,SPACE$(ln+sp)
  NEXT
NEXT
PRINT:PRINT"Premi un tasto."
WHILE INKEY$="":WEND
PRINT#2,CHR$(12);:ct=0
RETURN

Sillaba:
b$=UCASE$(LEFT$(ra$,s))
GOSUB 300:IF a=2 THEN s=s-1
GOSUB Elab
GOSUB 300:a$=x$
IF (a$<"A" OR a$>"Z") AND a<>2 THEN Sillaba
s=s+1:GOSUB 300
IF a$="S" AND NOT a AND x$<>"S" THEN s=s-2:RETURN
s=s-1:RETURN

Elab:
  IF s=0 THEN RETURN
  GOSUB 300
  IF a=2 THEN RETURN
  s=s-1
  IF a<>-1 THEN Elab
  GOSUB 300
  IF a=2 THEN RETURN
  IF a THEN Elab
  s=s-1:a$=x$:GOSUB 300
  IF a=1 THEN Elab
  IF s<2 THEN s=0:RETURN
  IF a THEN RETURN
  IF x$=a$ THEN RETURN
  IF x$="S" OR x$="G" THEN s=s-1:RETURN
  IF x$="N" OR x$="M" OR x$="R" OR x$="L" THEN RETURN
  IF a$="R" OR a$="L" OR a$="H" THEN s=s-1:RETURN
GOTO Elab
300 x$=MID$(b$,s,1)
    a=x$="A"OR x$="E" OR x$="I" OR x$="O" OR x$="U"
350 a=a OR x$="" OR x$="" OR x$="" OR x$="" OR x$="" OR x$=""
    IF NOT a AND (x$<"A" OR x$>"Z") THEN a=1
    IF x$=" " OR x$="-" OR x$="=" THEN a=2
RETURN
```

La subroutine **Stampa** serve, ovviamente, a stampare su più colonne il testo elaborato: ha a disposizione tutte le righe da stampare nel vettore **st$()** e le dispone in modo che nella prima colonna risultino le prime **tot** righe, nella seconda le righe da **tot + 1** a **tot x 2** e così via.

Il problema sta nel fatto che con una stampante convenzionale non si può posizionare il cursore casualmente nel foglio (come potrebbe avvenire per lo schermo), per cui grazie ad una banale formula (da cui ricaviamo la riga da stampare, data la colonna e la riga effettiva) ci limitiamo a stampare una riga dietro l'altra in senso orizzontale.

Nuova sfida

Il programma, testato in Redazione, è risultato il migliore (e, soprattutto, privo di gravi bug...) tra quelli inviati dai nostri lettori.

Tuttavia può esser migliorato ulteriormente sia completando la routine di sillabazione (in modo che risulti *realmente* universale), sia introducendo altri parametri, tra cui suggeriamo l'impostazione del numero di righe per pagina (ipotizzando l'invio del file verso una stampante).

Che altro suggerire?

Individuazione di caratteri "strani" (eventuali caratteri speciali non facilmente visualizzabili); "filtri" idonei alla loro interpretazione e sostituzione in base ad una tabella (vocali accentate, segni di tabulazione, eccetera).

La fantasia non ha limiti.

Se, poi, decidete di riscrivere il programma in **C** o in **Assembly**, beh, fatevi sentire...

Il regalo

All'autore del programma pubblicato in queste pagine, come promesso, è stato inviato un **pacco dono** contenente numerosi fascicoli di pubblicazioni Systems (riviste e dischetti).

✔ Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ HELP
❑ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
❑ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
✔ ARexx				
❑ Assembly				

Il linguaggio multitasking, proveniente da UNIX, è destinato a diventare uno standard su Amiga.

ARexx, iniziamo a conoscerlo

< di Ascanio Orlandini >

< Un nuovo linguaggio interprete per Amiga >

ARexx (**Amiga Rexx**) rappresenta l'implementazione su Amiga del linguaggio **Rexx**, diffusissimo da tempo su sistemi operativi UNIX e disponibile, nonostante la spiccata applicazione multitasking, anche sui "piatti" sistemi MS - DOS.

E' un linguaggio ad alto livello, creato appositamente per l'utilizzo di potenti **macro**, ma ottimo per qualunque tipo di programmazione che non richieda una elevata velocità elaborativa in quanto ARexx è un linguaggio esclusivamente **interpretato** e, come tale, non può competere con il **C**, ma nemmeno con altri linguaggi notoriamente lenti come i vari BASIC compilati.

Il linguaggio si dimostra particolarmente versatile anche come primo linguaggio da studiare: chi deve avvicinarsi per la prima volta in un ambiente di programmazione, generalmente seleziona il classico BASIC, che però si vede costretto ad abbandonare nel momento in cui decide di andare oltre, verso linguaggi più evoluti come il **C** oppure il **Pascal**.

Chi inizia, invece, a lavorare con ARexx, acquisisce un'impostazione mentale più vicina agli odierni linguaggi strutturati, grazie alla versatilità, potenza e maneggevolezza disponibile.

Al momento di abbandonare ARexx, cioè quando la velocità di esecuzione sarà veramente necessaria, potrà comunque essere utilizzato per la sua caratteristica, unica nel suo genere, di poter interagire con uno o più programmi, di mettere in comunicazione task diversi, di creare macro dalla potenza incredibile e scripts (o batch) veramente intelligenti.

Questo è senza dubbio uno dei più semplici e, allo stesso tempo, potenti linguaggi di programmazione oggi disponibili, al quale sarà facile cedere.

Non per nulla la Commodore ha deciso di includere ARexx insieme al **sistema operativo 2.0** al posto dell'obsoleto **AmigaBasic** Microsoft che ha accompagnato l'utente Amiga fin dall'inizio.

"Dentro" un programma

I programmi ARexx sono costituiti da normali testi ASCII, creati con l'editor preferito, ed eseguiti invocando direttamente l'interprete fornendogli, come argomento, il solo nome del programma da eseguire.

Per esempio, se vogliamo lanciare il programma dal nome **prova**, dovremo digitare...

```
rx prova
```

...in cui **rx** è il nome dell'interprete ARexx, disegnato per lavorare in ambiente CLI, anche se, volendo, è possibile attribuire ai programmi icone tramite **iconx**.

Il file programma può trovarsi sia nella directory corrente che nella directory **Rexx:** in cui vanno tipicamente memorizzati tutti i programmi di tipo ARexx.

Altra consuetudine che suggeriamo è quella di porre il suffisso **.rexx** a tutti i programmi ARexx; suffisso che può essere omesso passandone il nome all'interprete (**prova.rexx**, ad esempio, viene eseguito anche con un semplice **rx prova**).

Addirittura, adottando la **W-Shell** al posto della normale Shell Commodore, è possibile lanciare programmi ARexx direttamente, come normali eseguibili, senza neppure usare il comando rx.

Sia ARexx, che la W-Shell, sono prodotti della **Wishful Thoughtful Development Corp** al prezzo suggerito di **50$** ciascuno, più che accessibile, quindi, per entrare in possesso di un potentissimo linguaggio di programmazione.

Sequenza	Priorità	Definizione
~	8	NOT logico
+	8	Prefisso positivo
-	8	Prefisso negativo
**	7	Elevamento a potenza
*	6	Moltiplicazione
/	6	Divisione
%	6	Divisione intera
//	6	Resto in divisione
+	5	Somma
-	5	Differenza
||	4	Concatenazione (senza spazio)
(spazio)	4	Concatenazione (con spazio)
==	3	Esattamente uguale
~==	3	Esattamente diverso
=	3	Uguale
~=	3	Diverso
>	3	Maggiore
>=, ~<	3	Maggiore o uguale
<	3	Minore
<=, ~>	3	Minore o uguale
&	2	AND logico
|	1	OR logico inclusivo
^, &&	1	OR logico esclusivo

Tabella degli operatori

Dentro il linguaggio

Abbiamo già accennato al fatto che i programmi ARexx sono composti in puro ASCII; aggiungiamo ora che non è necessaria alcuna formattazione particolare. Un programma è composto da molteplici **Token**.

Un token è costituito dal più piccolo insieme di caratteri interpretabile dal linguaggio: token può essere la variabile **Prova**, l'istruzione **exit** o l'operatore più (+); in pratica un Token è tutto ciò che è separato da caratteri bianchi (spazio, tab, newline, ecc). Differenziamo adesso i token nelle diverse categorie.

❑ ❒

Token di commento

Similmente al linguaggio C, i commenti vengono introdotti nei programmi sorgenti con i caratteri di barra e asterisco (/*) e vengono conclusi con la stessa coppia invertita (*/). Sono consentiti commenti nidificati del tipo `/* Commento /* commento nidificato */ commento */`. Come facilmente intuibile, tutto ciò che è contenuto all'interno di un commento viene ignorato dall'interprete.

Symbol Token

Ogni gruppo di caratteri **a-z, A-Z, 0-9** e alcuni caratteri (**.!?$_**) vengono tradotti in maiuscolo in fase di interpretazione in modo che un'ipotetica variabile *temp* non possa esser confusa con *TEMP*.

All'interno dei Symbol token troviamo diverse categorie:

Fixed: symbol token inizianti con una cifra (**0-9**) o con un punto (.);
esempio: `532.263`

Simple: symbol token che non iniziano con una cifra e non contengono punti;
esempio: `Prova`

Stem: symbol token che hanno un punto (.) alla fine;
esempio: `Nome.`

Compound: symbol token contenente uno o più punti al suo interno;
esempio: `Nome.Numero`

String Token: Insieme di caratteri iniziante e terminante con una coppia di apici (') o di doppi apici ("). Se questi devono essere inseriti in una stringa, basta ripeterli due volte.

Stringhe immediatamente seguite da una **x** oppure una **b** indicano, rispettivamente, che le stringhe rappresentano numeri esadecimali o binari e devono quindi contenere caratteri **0-9**, **A-F**, **a-f** per le prime (esadecimali); solo **0** e **1** (binari) per le seconde. Esempi:

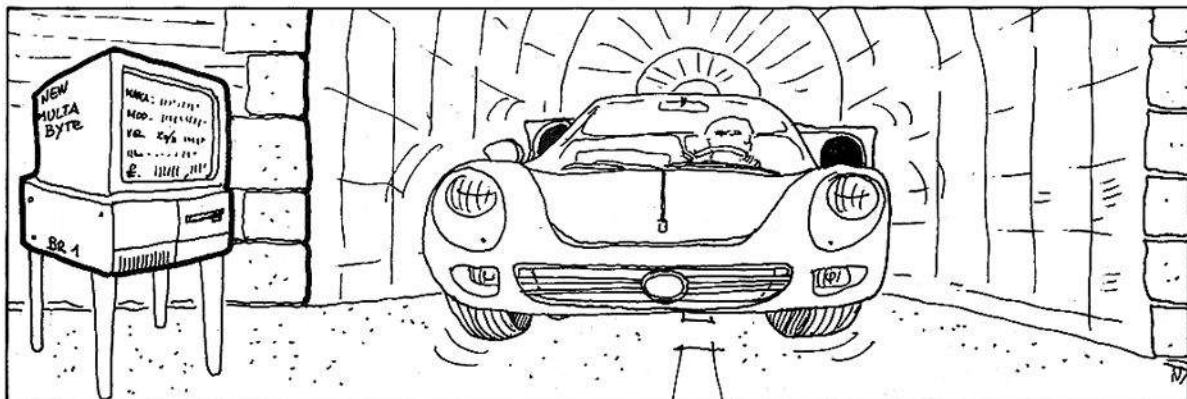
`"Questa è una stringa"`
`'Anche questa è una stringa'`
`'Torno all''alba'` equivale a `Torno all'alba`
`'0b 13 cf'x` è una stringa esadecimale
`'01101011'b` è una stringa binaria

Operator Token: Vari caratteri (~+-*/=&|^) possono essere combinati in sequenze differenti, come mostrato in tabella, per formare degli operatori (operator token) che possono essere preceduti, seguiti o avere internamente spazi (che vengono ignorati). Inoltre l'operatore **Blank** (ossia lo spazio) è considerato come operatore di concatenazione se segue un symbol e non è adiacente ad un operatore. Attenzione va posta alla diversa priorità degli operatori stessi: quelli con maggiore priorità sono eseguiti prima degli altri, salvo eventuali parentesi. Riferirsi sempre alla tabella.

❑ ❒

Caratteri speciali

⇨ **Doppio punto** (:): se preceduto da un symbol token determina quella che, in BASIC, è definita etichetta, ossia un punto di riferimento all'interno del programma, sfruttabile in diverse occasioni.

⇨ **Parentesi tonde** ( ): sono impiegate sia per alterare l'ordine di priorità degli operatori all'interno di espressioni, sia per identificare la chiamata ad una funzione; un symbol o una stringa seguita da una parentesi aperta è considerata come nome della funzione; all'interno

Dizionarietto

Token
Letteralmente *gettone*. Va inteso, nel contesto ARexx, come la parte più piccola di un programma interpretabile. Un token può essere una parola, un numero o un segno. (esempio: "CIAO", 123.2, +).

Symbol (Token)
Letteralmente simbolo, *segno*. In ARexx, un *symbol token* è un token costituito da caratteri alfanumerici, (lettere e / o numeri) più i caratteti punto (.) esclamativo (!) interrogativo (?) dollaro ($) sottolineato (_). Si divide in sotto-categorie.

Fixed (Symbol Token)
Letteralmente: *fisso*. Un fixed sysmbol token in ARexx è, in pratica, un numero con o senza parte decimale.

Simple (Symbol Token)
Letteralmente: *semplice*. Sostanzialmente in ARexx è una variabile normale, oppure un comando. In pratica, tutto ciò che non inizia con un numero e non contiene un punto all'interno.

Stem (Symbol Token)
Letteralmente: gambo, tronco, ramo, *radice*. Sono *stem* i token con un punto alla fine. In pratica sono le basi, le radici, dei token compound, come vedremo più avanti trattando di queste specie di array o matrici.

Compound (Symbol Token)
Letteralmente: *composto*. I compound symbol assomigliano concettualmente agli array del C o alle matrici del Basic. Sono insieme di caratteri alfanumerici (inizianti per carattere alfabetico) che hanno dei punti al loro interno.

String (Token)
Letteralmente: spago, *serie*. Sono le solite stringhe, ossia gruppi di caratteri delimitati, in questo caso, da apici o doppi-apici, che non vengono interpretati, ma presi così come sono.

Operator (Token)
Letteralmente: *operatore*. In effetti sono gli operatori (matematici e logici) costituiti da tilde (~), più (+), meno (-), per (*), diviso (/), eguale (=), maggiore (>), minore (<), et (&), barra verticale (|), elevazione a potenza (^).

delle parentesi di una chiamata a funzione ci sono i parametri passati alla funzione stessa.

`( 3 + 5 ) * 4` è un esempio del primo caso;

`error (3, 'I/O')` è un esempio di chiamata alla funzione error.

➪ **Punto e virgola** (;): equivale ad un terminatore di dichiarazione. Generalmente ogni singola linea contiene un'unica dichiarazione (terminata quindi da un line-feed). Più dichiarazioni possono essere consecutive, se separate da un punto e virgola (;). Ad esempio:

```
i=8; c=3; f='ciao';
```

➪ **Virgola** (,): la virgola viene usata quando un'unica dichiarazione non è contenuta su un'unica riga e deve continuare su una nuova linea; la virgola indica che la riga prosegue su quella successiva, come se si trattasse di un'unica, lunga linea.

Conclusioni

In queste pagine abbiamo dato uno sguardo ad ARexx in generale, ed ai Token in particolare, per lasciarvi il tempo di procurarvi il linguaggio in confezione originale (il basso prezzo limita, di fatto, il ricorso a copie pirata, tanto più che prossimamente ci riferiremo spesso al manuale...).

In caso di difficoltà, nel reperimento del programma, ci si può rivolgere telematicamente alla Bbsystem gestita dall'autore di queste note (02 / 57605211) oppure, più velocemente, presso The Gold Dragon BBS (0373 / 86966) per effettuare anche importazioni dagli USA.

✔ Amiga	❑ Principianti	✔ Esperti	❑ Tutti	✔ HELP
❑ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
❑ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
✔ C				
❑ Basic				
❑ Assembly				

Un breve programma, ampiamente commentato, vi consente di approfondire un argomento fondamentale.

Uno sguardo alle librerie "C"

< di Luigi Callegari >

< Abbiamo iniziato a parlare del linguaggio "C" per Amiga fin dal N. 81!>

Lo scorso mese abbiamo esaminato i concetti fondamentali di **finestra** e **schermo** in Amiga.

Ovviamente queste sono soltanto le basi di partenza per la realizzazione di programmi che sfruttano la **grafica**. In effetti, il sistema operativo grafico di Amiga, con le sue librerie **Graphics** ed **Intuition** (tralasciando la Layers di *basso livello*), è talmente complesso che la sua trattazione appena approfondita, effettuata dalla Commodore stessa nei famosi **R**om **K**ernel **M**anuals (RKM), occuperebbe dozzine di pagine, senza contare i lunghi listati necessari per adeguati esempi.

Per i listati, i volenterosi possono rivolgersi sia a quanto riportato nei RKM, sia nell'oramai abbondante circuito del pubblico dominio per Amiga.

Il linguaggio C, infatti, è da sempre principe per tutti i dilettanti programmatori di Amiga e si contano pertanto una infinità di listati liberamente studiabili e manipolabili (solitamente, però, in lingua inglese).

In queste pagine vedremo di dare accenni, spazio permettendo, sui concetti e sull'utilizzo pratico delle funzioni di base in brevi listati. In futuro stiamo già pensando a nuovi modi, di cui ovviamente i nostri lettori saranno informati tempestivamente, per fornire materiale di studio e sperimentazione a tutti gli apprendisti programmatori.

❑ ❑

Grafica e librerie

Prima di usare le librerie di Amiga, contenenti le eredi delle subroutine che si usavano con VIC/20 e C/64 (ora chiamate in Amiga **funzioni**) bisogna sapere almeno che cosa sono.

Ciascuna libreria contiene una serie di indirizzi di salto, uno per ogni subroutine in ROM. Dal momento che vi sono vari **Kickstart** (1.2, 1.3 e 2.0) per Amiga, l'uso delle librerie si rivela indispensabile.

Infatti, ogni Kickstart dispone di una propria area di memoria ove giace, ad un indirizzo ovviamente ben definito, **variabile a seconda delle versioni**, una ben precisa funzione. Gli **indirizzi** delle funzioni sono contenuti nelle **librerie**: ciò che il programmatore o, nel caso del linguaggio C, il compilatore conosce è l'indirizzo di tutti i **vettori** (ovvero, di istruzioni di salto) che, essendo di dimensioni fisse (sono istruzioni di salto per il processore 68000), non variano.

E' **Exec**, il cuore del sistema operativo, che si occupa di caricare la **tabella dei vettori** in un modo che risulta accessibile a **tutti** i programmi e che, contemporaneamente, risulta aggiornato in funzione dell'indirizzo assoluto delle funzioni in memoria per il Kickstart attualmente caricato.

Aprire la libreria

Volendo partire dalla grafica, per aprire la sua libreria, chiamata **graphics.library**, si usa la coppia di istruzioni del tipo indicato nel piccolo riquadro di questa pagina.

La funzione **OpenLibrary()** appartiene a **Exec**, che essendo la libreria di siste-

```
struct GfxBase *GfxBase;
GfxBase = OpenLibrary ("graphics.library",0);
```

Le due istruzioni indispensabili per evitare la comparsa del Guru.

```
struct IntuitionBase *IntuitionBase;
IntuitionBase = OpenLibrary("intuition.library",0);
```

Istruzioni necessarie per aprire correttamente la libreria di Intuition.

ma è sempre aperta. Restituisce un valore, un indirizzo, che occorre per inizializzare la variabile globale **GfxBase**. Tale variabile assume un significato ben preciso per i compilatori Amiga, essendo destinata a contenere, appunto, l'indirizzo della libreria grafica aperta da OpenLibrary(), che internamente viene definita come una struttura (**struct GfxBase**).

Si noti che per questo motivo **non** è consentito usare un nome differente ed è **obbligatorio** inserire le due linee riportate in **tutti** i programmi che usano funzioni grafiche.

In caso contrario, il compilatore genererebbe codice senza problema, ma al momento dell'esecuzione il codice compilatore chiamerebbe il Guru, tentando di balzare nelle funzioni della libreria senza avere l'indirizzo giusto dove si aspetta che sia.

Analogamente, tutti i programmi che usano funzioni della libreria di **Intuition** devono avere due linee del tipo indicato nel riquadro in alto in questa pagina per inizializzare il puntatore globale IntuitionBase.

Lo **zero** che si specifica come secondo parametro della **OpenLibrary()** indica che si richiede una versione *qualunque* di Kickstart; specificando, ad esempio, **33** si richiederebbe invece l'apertura della libreria della **V1.3** (e la funzione restituirebbe un **Null**, invece di un puntatore valido, come quando si verifica qualche problema per il quale non è riuscita l'apertura).

In effetti, per evitare che il compilatore generi un **warning** in fase di compilazione, bisognerebbe introdurre anche dei **cast** nelle linee, ma si tratta di una finezza, che però consente al compilatore di verificare che si stia agendo correttamente (vedi riquadro in basso).

In ogni caso, anche tralasciando il cast, il compilatore genera un codice corretto e funzionante, sebbene produca in compilazione un warning per indicare che ha dovuto eseguire automaticamente una conversione, non esplicitamente richiesta dal listato e quindi potenzialmente non voluta dal programmatore.

View, ViewPort e RastPort

L'intero schermo di Amiga è considerato una **Vista**, in inglese **View**, suddivisibile in più **ViewPort** gestibili separatamente dal sistema operativo.

La struttura View interna di Intuition punta effettivamente alla **prima** di varie strutture ViewPort, gestite automaticamente dal sistema e concatenate tra loro tramite puntatori interni. Ciascuna viewport, pure essa una struttura di Intuition facilmente manipolabile da C, specifica una propria risoluzione, dei colori ed eventuali caratteristiche speciali, limitatamente alle possibilità hardware.

Tutti sappiamo, ad esempio, che si possono avere più schermi sovrapposti con differenti modalità (facendoli scorrere col mouse uno sopra l'altro).

Ad ogni ViewPort è associata un'area grafica detta **Raster** rappresentata materialmente da una struttura BitMap, che può raggiungere i 1024 x 1024 pixel. Tale struttura descrive via software l'area grafica di lavoro mentre la struttura **RastPort** descrive il modo in cui le routine delle librerie grafiche e di Intuition devono trattare i dati contenuti nella struttura BitMap.

Difatti, ogni volta che si desidera utilizzare una funzione grafica per tracciare una linea o scrivere del testo, bisogna specificare un puntatore alla rastport associata all'area di lavoro interessata.

Il sistema operativo usa comunque un gran numero di strutture, per così dire "inferiori", la cui maggior parte può esse-

```
GfxBase=(struct GfxBase*)OpenLibrary("graphics.library,0);
IntuitionBase=(struct IntuitionBase*)OpenLibrary("intuition.library",0);
```

Le linee necessarie per evitare segnalazioni di Warning in fase di compilazione.

re ignorata tranquillamente dagli apprendisti programmatori.

❑ ❒

I colori

Denise, il chip custom che in Amiga sovraintende la gestione della grafica, dispone di 32 registri hardware per i colori.

Dal punto di vista del software, tali colori sono chiamati **Penne** (dall'inglese **Pen**) e sono gestiti dinamicamente dal sistema operativo. La penna numero **zero** contiene tipicamente il colore usato per lo sfondo (**background**), essendo il colore utilizzato quando tutti i bit dei bitplane associati ad una porzione di schermo assumono lo zero logico.

Normalmente, si usa un gruppo di penne per tracciare i grafici: una o più per il **foreground** (o **primo piano**), una per il **background**, una o più per il riempimento (**Fill**) di aree e così via.

Se si utilizza la penna numero **uno** per tracciare una linea e nel registro hardware di Denise è associato, per quella viewport, il colore **rosso**, la linea apparirà rossa.

Se però, in seguito, si modifica il contenuto di tale registro (non direttamente, in linguaggio C, ma scrivendo nella struttura di controllo ViewPort tramite apposite funzioni o macro di libreria grafica), **quella** retta e **tutte** quelle tracciate con la penna uno assumeranno il nuovo colore (il fenomeno di **ciclying**, ben noto anche a chiunque abbia usato il tasto Tab in Deluxe Paint, ad esempio).

Le funzioni che modificano il colore di tracciatura sono tre:

```
SetAPen (RP, penna);
SetBPen (RP, penna);
SetOPen (RP, penna);
```

...dove **RP** è un puntatore alla rastport della finestra interessata e **penna** è il numero di penna da usare, rispettivamente per le tre funzioni, come foreground, background e **outlining** (per i dettagli delle immagini).

Esistono funzioni che permettono di modificare il colore associato ad una penna, usando convenzionalmente un codice che specifica i sedicesimi di tinta fondamentale (rosso, verde o blu) per formare le 4096 sfumature consentite da Denise. La funzione più importante è:

```
SetRGB4( ViewPort, R, V, B );
```

...dove **ViewPort** è il puntatore alla viewport di cui si vogliono modificare le associazioni penne-colori effettivi e i tre numeri **R**, **V** e **B** specificano (in sedicesimi) le *dosi* di tinte fondamentali da mescolare.

Ovviamente, il massimo numero di penne dipende dal massimo numero di bitplanes...

1 bitplane	2 penne
2 bitplane	4 penne
3 bitplane	8 penne
4 bitplane	16 penne
5 bitplane	32 penne

...nel modo video standard (tralasciano HAM EHB ed i nuovi modi permessi dallo ESC montato di serie su Amiga 3000).

Sono dunque permesse attualmente, nei modelli **senza** ECS, 32 colori in bassa risoluzione e 16 colori in alta risoluzione (interlacciato).

Per ricavare l'indirizzo della viewport associata ad una finestra si usa una funzione specifica...

```
VP =ViewPortAddress (Window);
```

...dove **VP** è la variabile destinata a contenere un puntatore a struttura ViewPort e **Window** è il puntatore alla struttura della finestra interessata (il parametro restituito da OpenWindow(), insomma).

Esiste anche una funzione che consente di leggere interamente o parzialmente una serie di valori da assegnare alla tavolozza dei registri di Denise:

```
LoadRGB4(ViewPort, ColorMap, Penne);
```

...dove **ViewPort** è il puntatore alla viewport interessata, **ColorMap** è il puntatore alla matrice di valori a sedici bit (Uword) che definiscono il colore nel formato 0XRGB (rosso, verde e blu in sedicesimi) e **Penne** indica quanti valori devono essere letti nella matrice (al massimo 32).

❑ ❒

Disegnare

La libreria di Amiga comprende una quantità di funzione e macrodefinizioni per tracciare linee, rette, poligonali, circonferenze ed ellissi in modo rapido e relativamente semplice.

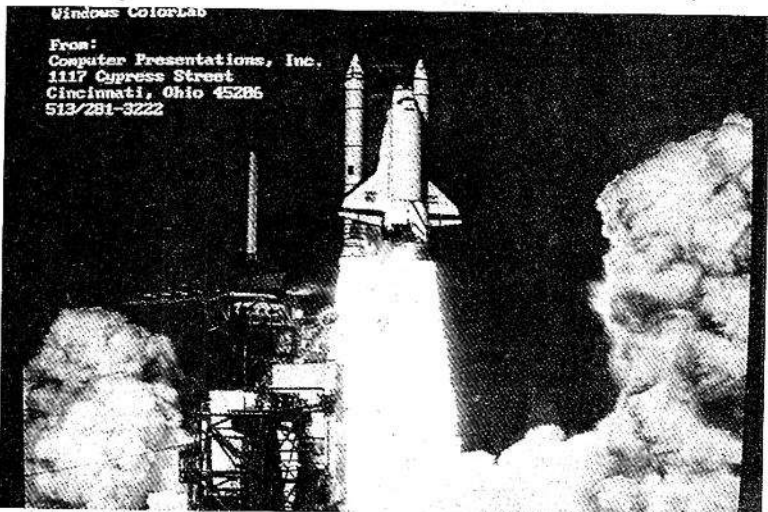
Elenchiamo qui alcune delle funzioni basilari, partendo innanzitutto da...

```
Move ( RP, x, y );
```

...che colloca sulla finestra di cui si specifica la rastport tramite puntatore (**RP**) alle coordinate assolute **x** ed **y** riferite allo spigolo superiore sinistro.

Nel movimento, non viene lasciata alcuna traccia, come è invece compito di...

```
Draw ( RP, x, y );
```

...che, appunto, sposta il cursore grafico alla posizione assoluta (x, y) riferita allo spigolo superiore sinistro della finestra di cui si specifica la rastport lasciando la traccia di colore specificato dalle penne di foreground e background e dal modo di tracciatura attualmente assegnato alla rastport (vedi riquadro).

Modi di tracciatura

Nel file di inclusione **graphics / rastport.h** vengono definite alcune costanti che consentono al programmatore di specificare come devono essere utilizzati i colori delle penne di foreground (primo piano) e di background (sfondo) mentre si scrive o si disegna in una finestra.

Ricordiamo che con le funzioni **SetAPen()** e **SetBPen()** illustrate in queste pagine si possono scegliere i colori delle penne tra i colori assegnati ai registri hardware di Denise. I quattro modi di tracciatura sono selezionabili con la funzione **SetDrMd(RP, modo)**, dove **RP** è il solito puntatore alla rastport e **modo** è una o più (sommati logicamente con il segno di barra verticale | oppure matematicamente con il segno di somma più +) delle seguenti costanti:

JAM1. Usa un solo colore, APen, per accendere i pixel di foreground, mentre il background resta inalterato.

JAM2. Usa APen per rappresentare un pixel acceso e BPen per rappresentare un pixel spento.

Complement. Inverte i pixel, cioè ogni pixel acceso viene spento e viceversa. Combinato con JAM2 (SetDrMd(rp, Complement + JAM2)) può essere usato per disegnare e cancellare.

Inversvid. Inverte i ruoli delle penne, usando APen per lo sfondo e BPen per il corpo dei caratteri o dei pixel accesi.

Più complicata, ma di poco, è la funzione...

```
PolyDraw (RP, coord, indir);
```

...che consente di tracciare una sagoma nella finestra di cui si specifica la RastPort (RP) unendo i punti le cui coordinate sono specificate in un vettore di **Short** o **Word**, di cui si specifica l'indirizzo come terzo parametro ed il numero di elementi come secondo parametro di PolyDraw(). Per disegnare un rettangolo eventualmente pieno di colore, si usa invece...

```
RectFill(RP,x1,y1,x2,y2);
```

...che accetta come parametri il puntatore alla rastport della finestra interessata (che contiene internamente le definizioni dei colori e dei modi di tracciatura, regolati con le apposite funzioni) e le coordinate assolute (relative allo spigolo superiore sinistro della finestra) di due spigoli opposti del rettangolo da tracciare. Esistono anche altre funzioni di tracciatura, come **AreaMove()**, **AreaDraw()**

La struttura RastPort

La struttura di una porta raster è definita nel file graphics/rastport.h come segue:

```
struct RastPort {
   struct Layer       *Layer;
   struct BitMap      *BitMap;
  USHORT             *AreaPtrn;
  struct TmpRas      *TmpRas;
  struct AreaInfo    *AreaInfo;
  struct GelsInfo    *GelsInfo;
  UBYTE              Mask;
  BYTE               FgPen, BgPen;
  BYTE               AOlPen, DrawMode;
  BYTE               AreaPtSz, dummy;
  BYTE               linpatcnt;
  USHORT             Flags, LinePtrn;
  SHORT              cp_x, cp_y;
  UBYTE              minterms[8];
  SHORT              PenWidth, PenHeight;
  struct TextFont    *Font;
  UBYTE              AlgoStyle, TxFlags;
  UWORD              TxHeight, TxWidth;
  UWORD              TxBaseLine, TxSpacing;
  APTR               *RP_User;
  ULONG              longreserved[2];
  UWORD              wordreserved[7];
  UBYTE              reserved[8];
};
```

...dove i campi più significativi per l'utente medio sono:

BitMap. Contiene il puntatore alla struttura BitMap di definizione della dimensione della mappa in Chip Ram, tipicamente inizializzata con InitBitMap().

AreaPtrn. Contiene il puntatore alla regione di memoria usata durante il riempimento di aree in RAM, ovvero l'indirizzo della definizione della sagoma usata da AreaEnd() e Flood() per il riempimento di aree grafiche. Tale blocco è tipicamente inizializzato con la macro SetAfPt() contenuta nel file graphics/gfxmacros.h.

TmpRas. Contiene il puntatore alla struttura TmpRas di controllo di un buffer supplementare usato per contenere porzioni di bitmap durante l'esecuzione interna di funzioni come AreaDraw(), AreaMove() e AreaEnd().

Mask. E' la maschera che specifica quali bitplane alterare durante la tracciatura grafica, definita tramite una assegnazione a membro di struttura o la macro SetWrMsk() definita in graphics/gfxmacros.h.

FgPen e **BgPen**. Sono i valori delle penne correnti da usare per i pixel di foreground e background rispettivamente. Si assegnano normalmente con SetAPen() e SetBPen().

DrawMode. Contiene il valore corrispondente al modo di tracciatura da usare, come stabilito normalmente con SetDrMd(), che influisce sul modo in cui vengono effettivamente usati i colori delle penne durante la resa grafica.

PenWidth, PenHeight. Sono rispettivamente la larghezza e l'altezza della penna corrente, in pixel.

Font. Puntatore alla struttura di definizione del testo TextFont, che sarà oggetto della prossima chiacchierata su Computer Club.

```
mioschermo=(struct Screen *)OpenScreen(...);
miafinestra=(struct Window *)OpenWindow(...);
```

Due semplici righe evitano problemi in fase di compilazione.

e **AreaFill()**, che però richiedono una gestione più complessa, avendo bisogno di aree di lavoro (memoria) personale che deve essere fornita e gestita dal programmatore.

Il programma DemoGraf

Ritenendo per esperienza personale che si impara spesso molto di più da un listato semplice, ben commentato e funzionante che da una marea di cenni teorici, presentiamo anche questo mese (come sempre) un programma in C commentato, pronto per essere compilato con **Lattice SAS/C** oppure **Aztec C**. In esso sono usate alcune delle funzioni fin qui citate e può servire come piattaforma di sperimentazione dei vari concetti appresi, oltre a chiarire alcune implicazioni pratiche degli stessi.

Inizialmente, si definiscono i puntatori globali di sistema **GfxBase** ed **IntuitionBase,** nonchè della finestra, schermo e rastport che useremo per il nostro esperimento.

Troviamo poi la definizione di un vettore **Colormap** contenente i codici di otto colori secondo la convenzione dei *sedicesimi di tinta* detta precedentemente. Si noti che la specifica **0x** indica che il nu-

```
/* DemoGraf.c - Demo funzioni grafiche
   By Luigi Callegari x CC - 04/09/91
*/

#include <intuition/intuition.h>
#include <graphics/gfxmacros.h>

#ifdef LATTICE
#  include <proto/intuition.h>
#  include <proto/graphics.h>
#  include <proto/exec.h>
#else
#  include <functions.h>
#endif

#define INOME "intuition.library"
#define GNOME "graphics.library"
#define RP    miafinestra->RPort

struct IntuitionBase *  IntuitionBase;
struct GfxBase *        GfxBase;
struct Window *         miafinestra;
struct Screen *         mioschermo;
struct ViewPort *       viewport;

/** Mappa dei colori per schermo **/
/***********************************/
USHORT colormap[ 8 ] = {
  /* Bianco, Rosso, Verde, Blu */
     0xfff, 0xf00, 0x0f0, 0x00f,
  /* Ciano, Viola, Giallo, Nero */
     0x0ff, 0xf0f, 0xff0,  0x000
};

/** Definizione schermo personale **/
/************************************/
struct NewScreen NewScreenDef = {
  0,0, 320,255, 3, 6,4, NULL,
  CUSTOMSCREEN,NULL,NULL,NULL,NULL
};

/* Definizione window personale **/
/**********************************/
struct NewWindow NewWindowDef = {
  0,10, 320,245, 2,1, CLOSEWINDOW,
  SMART_REFRESH + BORDERLESS +
  ACTIVATE + WINDOWCLOSE, NULL,
  NULL, (UBYTE*)"Grafica", NULL,
  NULL, 0,0,0,0, CUSTOMSCREEN
};

/* Funzione di uscita "pulita" */
/*******************************/
void die( int n )
{
  if (miafinestra) CloseWindow(miafinestra);
  if (mioschermo) CloseScreen(mioschermo);
  if (IntuitionBase) CloseLibrary(IntuitionBase);
  if (GfxBase) CloseLibrary(GfxBase);
  _exit( n );
}

/* Funzione principale */
/***********************/
void _main( void )
{
  register y;

/* Apriamo le due librerie di sistema */
  IntuitionBase = OpenLibrary( INOME, 33L );
  GfxBase       = OpenLibrary( GNOME, 33L );

/* Se fallita apertura librerie, finisce */
  if ( !IntuitionBase || !GfxBase ) die( 10 );

/* Ora s'apre lo schermo, finisce se non OK */
  mioschermo = OpenScreen( &NewScreenDef );
  if ( mioschermo == NULL ) die( 11 );

/* Assegnamo puntatore schermo a window */
  NewWindowDef.Screen = mioschermo;

/* Ora s'apre la finestra, finisce se non OK */
  miafinestra = OpenWindow( &NewWindowDef );
  if ( miafinestra == NULL ) die( 12 );

/* Troviamo la viewport della finestra */
  viewport = ViewPortAddress( miafinestra );

/* Assegnamo la tavolozza dei colori */
  LoadRGB4( viewport, &colormap[0], 8L );

/* Cambiamo i colori e tracciamo delle linee */
  SetDrMd( RP, JAM2 );
  for ( y=9 ; y<251 : y+=3 ) {
     SetAPen( RP, 5 );
     Move( RP, y/2, y); Draw(RP, 320, 8);
     SetAPen( RP, 7 );
     Move( RP, y/3, y); Draw(RP, 320, 120);
     SetAPen( RP, 4 );
     Move( RP, y/4, y);
```

mero è espresso in base esadecimale e che le tre cifre hex che seguono indicano le componenti Rossa, Verde e Blu del colore da ottenere.

Segue la struttura di definizione dello schermo e della finestra, già accennate nel **n. 87** di Computer Club.

La funzione **die()** accetta come input un numero intero che rappresenta un codice di errore da restituire al processo CLI che chiama il programma compilato (tramite _**exit()**). Il suo scopo è di chiudere la finestra, lo schermo e le librerie ordinatamente per lasciare il sistema in ordine al termine del programma ([illegible] chiude con un **die(0)**, per l'appunto) sia in caso di errori verificatisi durante l'esecuzione.

La funzione _**main()** è quella principale, eseguita quando si invoca da CLI il programma. Il significato del carattere di underscore (_) è già stato chiarito il mese scorso, ma rammentiamo che consente di risparmiare l'inserimento di codice supplementare da parte del linker, quando possibile per il tipo di programma. Si inizia aprendo le due librerie GfxBase e IntuitionBase, assegnando alle apposite variabili globali i puntatori restituiti da OpenLibrary(), o terminando tramite Die() in caso di errore.

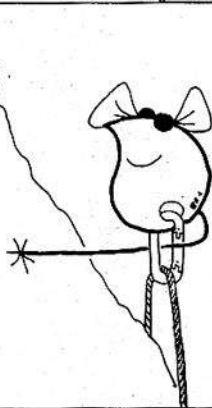
Si noti che, attualmente, usando Kickstart e Workbench **V1.3** l'unico remoto caso in cui si potrebbe verificare un errore nell'apertura delle librerie in ROM (a parte errori di digitazione del nome durante la battitura del listato...) è per una estrema ristrettezza di memoria, effettivamente difficile da verificarsi. Pur se praticamente obsoleto, è comunque buona pratica di programmazione inserire **sempre** questi controlli, così come le istruzioni di chiusura delle librerie aperte, anche se si trovano in ROM.

La **main()** prosegue aprendo lo schermo personale, passando a **OpenScreen()** un puntatore alla struttura di definizione riportata in testa al programma. Il compilatore dovrebbe generare un warning per questa assegnazione (SAS, Aztec è più permissivo) in quanto sarebbe necessario inserire un **casting** per convertire il puntatore restituito da OpenScreen (prima riga del riquadro di questa pagina) così come aprendo la finestra (seconda riga); ma, sapendo che il codice generato sarà comunque corretto (sappiamo già che la conversione implicita fatta dal compilatore del valore restituito da OpenWindow() ed OpenScreen() per eseguirne la memorizzazione nelle variabili avviene correttamente), possiamo anche evitare di appesantire il listato ed ignorare i brontolii di LC1.

Si noti che, prima di aprire la finestra, occorre assegnare nella sua struttura di definizione (**NewWindowDef**) il puntatore allo schermo aperto, a cui detta finestra appartiene.

Tale valore viene calcolato dal sistema dinamicamente, ovvero al momento dell'esecuzione del programma (si tratta di un indirizzo assoluto in memoria, che può essere sempre diverso) e non può quindi essere noto a priori e venire riportato nella struttura di definizione della finestra.

Dal momento che vogliamo usare **LoadRGB4()** per caricare la mappa dei colori definita nel vettore globale colormap(), dobbiamo ricavare l'indirizzo della viewport, pertanto usiamo la funzione **ViewPortAddress()** per assegnare alla variabile viewport tale indirizzo, in seguito passato alla funzione LoadRGB4() insieme all'indirizzo del vettore contenente le definizioni dei colori ed il numero degli stessi (8).

Poi segue un semplice ciclo che traccia tre serie di rette con differenti colori di foreground in modo JAM2. Infine, la funzione Delay() della libreria AmigaDOS attende 8 secondi (in cinquantesimi), prima di terminare l'esecuzione con la chiamata a die().

Per compilare e linkare il programma si può usare con SAS/C V5 una linea Shell del tipo:

```
LC -vbr -O -Lntv -ms Demograf.c
```

...oppure con Aztec C V5 le due linee:

```
CC -sb -so Demograf.c
LN Demograf -lc
```

...ignorando pure i warning innocenti prodotti dai compilatori. Il programma eseguibile così ottenuto dovrebbe occupare meno di 2 Kbyte.

✔ Amiga	❑ Principianti	✔ Esperti	❑ Tutti	✔ HELP
❑ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
❑ Applicazioni				
✔ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
✔ Assembly				

Come spsotare dati tra locazioni di memoria e registri.

L'istruzione Move e le sue varianti

< di Luigi Callegari >

< Esaminare C. C. n. 86, n. 87 >

Nella scorsa puntata abbiamo iniziato a parlare del set di istruzioni del processore 68000 descrivendo i vari modi di indirizzamento e la istruzione **Move**, sicuramente la più usata ed importante.

In effetti, l'istruzione Move ha alcune sorelle gemelle, riportate in queste pagine insieme alla capostipite per riferimento, la cui sintassi risulta praticamente identica e le differenze molto sottili.

Alcune di queste istruzioni sono completamente ignorate dai programmatori, in quanto vengono convertite automaticamente dagli assemblatori per Amiga. In pratica, dato che le differenze di queste istruzioni rispetto alla capostipite Move consistono nel **tipo** degli operandi, il programmatore può specificare sempre Move, sapendo che provvederà poi il programma assemblatore ad usare internamente le corrette MoveA, Move o, talvolta, anche MoveQ.

Facendo un esempio, possiamo dire che quando il programmatore scrive e compila una linea tipo...

```
MOVE.L #Lidia,A0
```

...indica che vuole copiare, nel registro **A0**, l'indirizzo in memoria dell'etichetta Lidia.

In effetti, però, dato che la destinazione è un registro indice (A0), la forma più corretta è:

```
MOVEA.L #Lidia,A0
```

...dal momento che è l'istruzione **Move Address** che sposta dati in un registro indice.

Difatti, tutti gli assemblatori standard per Amiga convertiranno la *prima* linea nella *seconda*, come si potrebbe vedere chiedendo il listato al momento della compilazione oppure disassemblando quanto assemblato. In modo simile, ma leggermente differente, una linea del tipo:

```
MOVE.L #-1,D0
```

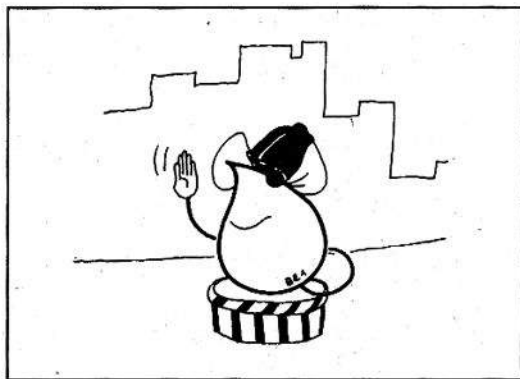
...potrebbe venire convertita più efficientemente nella seguente:

```
MOVEQ #-1,D0
```

L'istruzione **Move Quick**, infatti, occupa meno spazio in memoria e viene eseguita più velocemente di una Move normale.

Altera sempre i 32 bit del registro di destinazione (estendendo il bit di **segno**), perciò è superfluo specificare **.L**, ed è impossibile specificare **.B** oppure **.W** o un dato immediato la cui memorizzazione richieda più di 8 bit (con segno). Alcuni assemblatori, come **GenAm2** del Devpac, possono opzionalmente eseguire automaticamente anche questa ottimizzazione al momento dell'assemblaggio.

Si noti che alcune istruzioni possono essere eseguite solo in **modo Supervisore**, non nel normale **modo Utente** nel quale vengono normalmente eseguiti i programmi su Amiga.

I due modi di funzionamento sono illustrati solo brevemente, essendo legati a concetti e cognizioni sui microprocessori piuttosto profonde, che saranno oggetto di futuri appuntamenti.

MOVE

Sintassi: MOVE.d <ea1>,<ea2>

Sorgente: Dn, An, (An), (An)+, -(An), (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L, #<dato>, (d16,PC), (d8,PC,Xn), (bd,PC,Xn), [(bd,PC,Xn),od], [(bd,PC),Xn),od].

Destinazione: Dn, An, (An), (An)+, -(An), (d16,An), d8,An,Xn), (bd;An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L, #<dato>, (d16,PC), (d8,PC,Xn), (bd,PC,Xn), [(bd,PC,Xn),od], [(bd,PC),Xn),od].

Funzione: Copia dati da un indirizzo effettivo ad un altro. La dimensione può essere B, W od L.

Flag: X non alterato, N=1 se il risultato è negativo, Z=1 se il risultato è zero, V e C sempre azzerati.

Esempio:

```
MOVE.L D0,D1
```

Effetto: Copia i 32 bit del registro D0 nel registro D1.

Note: Può essere convertita automaticamente in MOVEA o MOVEQ (vedere).

MOVEA

Sintassi: MOVEA.d <ea>,An

Sorgente: Dn, An, (An), (An)+, -(An), (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L, #<dato>, (d16,PC), (d8,PC,Xn), (bd,PC,Xn), [(bd,PC,Xn),od], [(bd,PC),Xn),od].

Destinazione: An.

Funzione: Copia dati da un indirizzo effettivo in un registro indice, lasciando i flag inalterati. La dimensione può essere W od L, ma il segno viene esteso comunque a 32 bit nella destinazione.

Flag: Tutti inalterati.

Esempio:

```
MOVEA.W D0,A0
```

Effetto: Copia i 16 bit meno significativi del registro D0 nel registro A0.

Note: L'istruzione Move con destinazione un registro indice viene convertita da tutti gli assemblatori standard di Amiga in MoveA.

Move to CCR

Sintassi: MOVE <ea>,CCR

Sorgente: Dn, (An), (An)+, -(An), (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L, #<dato>, (d16,PC), (d8,PC,Xn), (bd,PC,Xn), [(bd,PC,Xn),od], [(bd,PC),Xn),od].

Destinazione: CCR

Funzione: Copia gli otto bit inferiori dell'operando effettivo negli otto bit inferiori del registro di stato CCR. L'istruzione ha sempre dimensione word, ma gli 8 bit superiori del sorgente vengono ignorati e gli 8 bit superiori rimangono inalterati.

Flag: X vale il bit 4 dell'operando, N il bit 3, Z il bit 2, V il bit 1 e C il bit 0.

Esempio: MOVE DO,CCR

MOVE from CCR

Sintassi: MOVE.s CCR,<ea>

Sorgente: CCR (registro interno di stato)

Destinazione: Dn, (An), (An)+, -(An), (d16, An), d8, An, Xn), (bd, An, Xn), ([Bd,An,Xn],od), xxx.W, xxx.L.

Funzione: Copia gli 8 bit inferiori dello Status Register in un indirizzo effettivo, lasciando i flag inalterati.

Flag: Tutti inalterati.

Esempio:

```
MOVE CCR,D0
```

Effetto: Copia il CCR nei 16 bit inferiori del registro D0.

Note: Questa istruzione corregge, nelle CPU successive al 68000, un bug. In questo processore, infatti, i programmi in modo utente potevano leggere il CCR (compreso il byte superiore) e non vi era istruzione per leggere soltanto i flag. Per questo scopo, molti programmi usavano MOVE from SR, ma aggiungendo (a partire dal 68010) MOVE from CCR e rendendo MOVE to SR una istruzione privilegiata, si è evitata la possibilità da parte dei programmi utente di leggere il byte di sistema del registro di stato. Per contro, il 68000 risulta tecnicamente incompatibile, sotto questo aspetto, con il 68010 ed i chip superiori.

MOVE from SR

Sintassi: MOVE SR,<ea> (privilegiata)

Sorgente: Status Register

Destinazione: Dn, (An), (An)+, -(An), (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L.

Funzione: Copia il registro di stato in un indirizzo effettivo, lasciando i flag inalterati. E' una istruzione privilegiata su tutti i processori, ad eccezione del 68000, perciò se eseguita in modo utente da 68010/20/30 provoca una eccezione per violazione di privilegio.

Flag: Tutti inalterati.

Esempio:

```
MOVE SR,D0
```

Effetto: Copia nei 16 bit inferiori di D0 il contenuto del registro di stato.

MOVE to SR (privilegiata)

Sintassi: MOVE <ea>,SR

Sorgente: Dn, (An), (An)+, -(An), (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L, #<dato>, (d16,PC), (d8,PC,Xn), (bd,PC,Xn), [(bd,PC,Xn),od], [(bd,PC),Xn),od].

Destinazione: Status Register.

Funzione: Copia i 16 bit dell'indirizzo effettivo nel registro di stato, alterando appropriatamente i flag.

Flag: X=bit 4 dell'operando sorgente, N=bit 3, Z=bit 2, V=bit 1, C=bit 0.

Esempio:

```
MOVE D0,SR
```

Effetto: Copia nel registro di stato i 16 bit inferiori del registro dati D0.

MOVE USP (privil.)

Sintassi:

Move USP,AN

Move An,USP

Sorgente: An od USP.

Destinazione: An od USP.

Funzione: Consente ad un programma supervisore di ricavare e fissare lo stack pointer del modo utente. In pratica, questa istruzione consente ai programmi in modo supervisore di inizializzare, salvare e ripristinare l'area di stack dei programmi eseguiti in modo utente. E' vitale per il supporto del multitasking. I 32 bit dello stack pointer (A7) possono venire scritti in un registro indice dopo essere stati letti da un registro indice.

Flag: Tutti inalterati.

Esempio:

```
MOVE USP,A0
```

Effetto: Copia in A0 il contenuto di A7.

MOVEM

Sintassi:

MoveM.d <lista>,<ea>

MoveM <ea>,<lista>

Sorgente: Lista di registri o (An), (An)+, (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L, (d16,PC), (d8,PC,Xn), (bd,PC,Xn), [(bd,PC,Xn),od], [(bd,PC),Xn),od].

Destinazione: (An), -(An), (d16,An), d8,An,Xn), (bd,An,Xn), ([Bd,An,Xn],od), xxx.W, xxx.L.

Funzione: Trasferisce più registri da o verso la memoria. La lista di registri consiste nel loro nome separato da slash (/), oppure da una gamma interposta dal segno meno (-). Ad esempio: D0/D4-D6/A1 specifica i registri D0, D4, D5, D5 ed A1. La grandezza può essere W od L.

Flag: Inalterati

Esempio:

```
MOVEM.L D0/D1/A0/A1,(-A6)
```

Effetto: Memorizza in uno stack a discesa, puntato da A6, gli interi contenuti di D0, D1, A0 ed A1.

Nota: Quando si sposta dalla memoria ad un registro con dimensione word, MOVEM estende il segno e quindi influisce su tutti i 32 bit del registro destinazione.

MOVEP

Sintassi:

MOVEP d(Ay),Dx

MOVEP Dx,d(Ay)

Sorgente: Registro dati o indirizzo di memoria.

Destinazione: Registro dati o indirizzo di memoria.

Funzione: Sposta dati tra un registro dati e byte di memoria alternati, lasciando i flag inalterati.

Flag: Inalterati.

Esempio:

```
MOVEP 0(A0),D0
```

Effetto: Carica in D0 i byte ad indirizzi pari partendo dalle due longword puntate da A0.

Note: Questa istruzione è utile per il dialogo tra il 68000/68010 e vecchi chip a 8 bit, assistendo il processo di copia di 2 o 4 byte in un registro dati in successivi indirizzi in memoria ad indirizzi differenti di due unità. Nel 68020/30 esiste un dimensionamento dinamico del bus per il quale MOVEP risulta praticamente superflua.

MOVEQ

Sintassi:

MOVEQ #<dato>,Dx

Sorgente: Dato ad otto bit con segno (-128/+127).

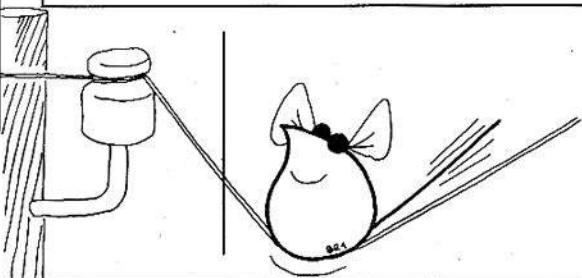
Destinazione: Registro dati.

Funzione: Scrive in un registro dati un valore immediato ad otto bit, occupando meno memoria della consueta MOVE e funzionando più velocemente. Il segno della sorgente viene sempre estesa ad alterare tutti i 32 bit del registro destinazione.

Flag: X non alterato, N=1 se il risultato è begativo, Z=0 se il risultato è zero, V et C sempre azzerati.

Esempio:

```
MOVEQ #-1,D1
```

Effetto: Scrive in D1 la costante -1, ovvero il numero $FFFFFFFF, attivando il flag N ed azzerando tutti gli altri.

Modo utente e modo supervisore

Molti microprocessori avanzati, come i processori dei grandi *mainframes*, dispongono di due modi di funzionamento.

I processori Motorola della serie 68000 hanno il modo Supervisore ed il modo Utente. Quando il **bit 13** del registro di stato è attivo, la CPU si trova in modo Supervisore (o **Privilegiato**), ovvero può eseguire qualunque istruzione. Quando invece il bit 13 è azzerato, la CPU si trova nel modo utente, ovvero nella circostanza di esecuzione tipica della maggior parte dei programmi scritti dall'utente.

In questo stato, il microprocessore **non** può eseguire quelle istruzioni che possono interferire con il funzionamento normale del sistema operativo. Ad esempio, tutte le istruzioni che alterano il byte superiore del registro di stato non possono essere eseguite in modo utente.

In modo supervisore si può passare al modo utente, ma **il contrario non è possibile**, in quanto l'utente **non** può porre in modo supervisore la CPU normalmente.

Pertanto l'unico modo per passare da modo utente a modo supervisore è tramite una delle **eccezioni**, che comprendono interruzioni da periferica, errori di bus, trappole utente e di sistema ed istruzioni illegali.

Quando si verifica una eccezione, l'esecuzione prosegue ad un indirizzo noto al sistema operativo in modo supervisore, e nel caso di Amiga ciò, in condizioni normali, significa la produzione di una coerente **Guru Meditation**. E' virtualmente impossibile per un programma utente passare al modo supervisore accidentalmente.

Nei processori 68020/30 il bit di flag 12 permette **altri due stati supervisore** esclusivi, chiamati stato **Master** (M=1) e stato **Slave** (M=0).

La loro trattazione teorica non è di grande interesse per i possessori di Amiga ed esula pertanto dallo scopo di questo articolo.

POSTAMIGA

(a cura di Domenico Pavone)

Linguaggi e non

Ultimamente si sente tanto parlare di Amiga ed Ms-Dos. Come si può iniziare a scrivere un programma dopo essere entrati nel Cli o Shell? Ho provato con Type e simili, ma niente di fatto! Inoltre non ho ancora capito se Ms-Dos e l'Assembler sono "simili".
(Emanuele G. - Venezia)

La domanda denota una certa confusione, del resto giustificabile quando ci si accosta per la prima volta all'affascinante mondo dell'informatica. Per di più con un computer come Amiga, certo non facile come i suoi predecessori. Ma vediamo di sciogliere almeno qualche dubbio.

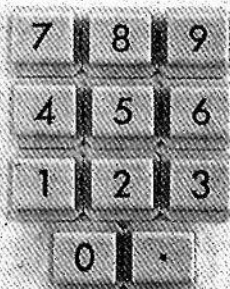
Intanto c'è subito da precisare che ogni computer ha un suo DOS (**D**isk **O**perative **S**ystem): Amiga ha il suo bravo **Amiga Dos**, mentre quello principalmente adoperato sugli Ibm compatibili si chiama **MS-DOS** (Ms sta per Microsoft, la ditta che lo produce). I due sono ovviamente diversi, ma in generale tutti i Dos (anche quelli di altri computer) si occupano della gestione dei dischi, dei file, della stampante, e altre amenità che non staremo qui a precisare.

Una volta chiarito questo punto, ecco in partenza un primo consiglio: se si possiede (solo) un Amiga e si è alle primissime armi, sarà opportuno **prima** imparare *almeno* i rudimenti del suo Dos e, solo in un secondo tempo, approfondire Ms-Dos, visto che quest'ultimo può essere adoperato su Amiga **unicamente** tramite l'uso di opportuni emulatori.

Tornando più direttamente alle domande: il Dos è dotato di precisi comandi, che nel caso di Amiga possono essere impartiti solo all'interno di una finestra Shell oppure Cli. L'insieme dei comandi, inoltre, può essere utilizzato come un vero e proprio linguaggio di programmazione: basterà creare un file di testo (chiamato **Batch File**) che contenga una sequenza dei comandi voluti. Il file di testo dovrà essere digitato utilizzando un editor che poi salvi il documento in puro formato ASCII. Il che significa: Type non c'entra niente, eventualmente (su Amiga) può andar bene **Ed**, se proprio non si è in grado di adoperare altri programmi specializzati.

Ma significa anche approfondire i comandi del dos e l'uso di Ed (o del text editor preferito), nonché sottostare a un imperativo cui un principiante (più degli altri) non può esimersi: consultare attentamente la manualistica fornita in dotazione al computer, aiutandosi poi con quanto ricavabile da pubblicazioni come la nostra. Per la cronaca, Amiga Dos è stato sviscerato a fondo in una serie di articoli apparsi dal numero **75** al numero **83**, eventualmente rintracciabili presso il nostro servizio arretrati.

Quanto appena detto è valido a grandi linee anche per l'Ms-Dos, solo che (come ovvio) i comandi saranno diversi (pur se esistono molte affinità), e comunque non adoperabili direttamente in Shell o Cli.

E veniamo all'ultimo dubbio: no, **Dos** e **Assembler** non sono affatto simili. Il primo, volendolo considerare come linguaggio, è dotato di

comandi piuttosto semplici e intuitivi, che possono svolgere compiti in un ristretto campo di azione.

L'Assembler, invece, è un vero linguaggio, ma molto più "vicino" al computer che al suo utilizzatore: quindi di estrema potenza, in grado di fare pressoché tutto, ma... terribilmente difficile, almeno per un principiante.

Il che non significa che non lo si capirà mai, intendiamoci, ma... ogni cosa a suo tempo. Amiga o Pc che sia, saperne sfruttare il Dos è un primo passo indispensabile per ogni futura evoluzione, anche se si dovesse decidere di rimanere un semplice "user" del computer.

Come lo salvo?

Posseggo un Amiga 500 da circa 6 mesi, e non sono ancora riuscito a capire come si fa a salvare dati, programmi, file, eccetera su un disco che non sia quello sul quale si lavora. Comincio a rimpiangere il mio vecchio C/128...

(Vincenzo Ienzo)

La domanda è ricorrente, ma una breve risposta è ugualmente d'obbligo, non foss'altro che per attenuare quel rimpianto destinato a scomparire in breve tempo, una volta presa confidenza con Amiga.

La soluzione al problema è molto semplice: basta citare, quando si fornisce il nome del file da salvare, **anche** il nome assegnato al disco nel quale si intende memorizzarlo.

In pratica: se per esempio si ha un disco a nome **Work**, *diverso* da quello in uso, e vi si vuole salvare un file con nome **Prog**, la stringa da adoperare sarà...

```
Work:prog
```

(senza omettere il carattere di doppio punto!).

Questo vale per Amiga Basic (come specificato in altra parte della lettera), ma anche per **qualunque altro programma** (word processor, data base, eccetera).

L'uso del nome del disco è particolarmente indicato quando si possiede il solo drive interno, mentre in presenza di una seconda unità è anche possibile specificare la periferica, per esempio...

```
Df1:prog
```

Come ovvio, con le stringhe appena viste il file verrà inserito nella directory principale del disco.

Se, invece, si intende adoperare una *subdirectory*, basterà specificarla nel path (si chiama così la descrizione del percorso ove è rintracciabile un file). Per esempio:

```
Work:subdir/prog
```

In questo caso, deve naturalmente essere già presente una directory di nome **Subdir** nel disco Work.

Non è questione di gusti

Come mai Deluxe Paint setta la risoluzione verticale a 200 oppure 400 linee e voi, nei vostri programmi, a 256 oppure 512?

(Alberto Molisani)

Il perché della differenza la si può scoprire proprio dando un'occhiata all'ultima versione di Deluxe Paint, che nella schermata iniziale propone la scelta tra **Pal** e **Ntsc**.

Il primo è uno standard **europeo**, il secondo **americano**.

Il Pal, adottato da tutti i nostri monitor, consente una risoluzione, per l'appunto, di 256 pixel (**non** righe) in verticale, che in modo interlacciato diventano 512. L'Ntsc, invece, è limitato a 200 e 400 pixel verticali.

Anche la versione precedente di Deluxe Paint comunque, nella sua versione europea, consentiva già di disegnare a tutto schermo.

Visto che si dispone di tanto spazio in più, perché non sfruttarlo?

Sviluppatori Amiga, unittevi!

Sabato 2 Novembre a **Modena**, presso la sala riunioni del nuovo Planetario, si terrà IPISA 91 (**I**ncontro **P**rogrammatori **I**taliani **S**vilupatori **A**miga). Si tratta di una occasione unica per gli sviluppatori ufficiali Commodore, e per gli aspiranti tali, di incontrarsi e di incontrare i rappresentanti della Commodore. La disponibilità di posti è limitata ed è necessaria la prenotazione. La quota di lire **18000** comprende anche alcuni gadgets di notevole valore quali un disco di programmi PD sviluppati in Italia ed un libro, pubblicato espressamente per l'occasione. Gli interessati possono contattare il Andrea Salati, Viale Jsaac Newton 25, 41100 Modena, Tel. 059/331467 (oppure indirizzargli un Matrix al 2:332/505.6 @ fidonet.org).

Stessa icona, file diversi

Vorrei sapere come fare per collegare una icona diversa dalla sua a un file oppure a una directory. Da Workbench non si riesce!

(Silvio Manucci - Firenze)

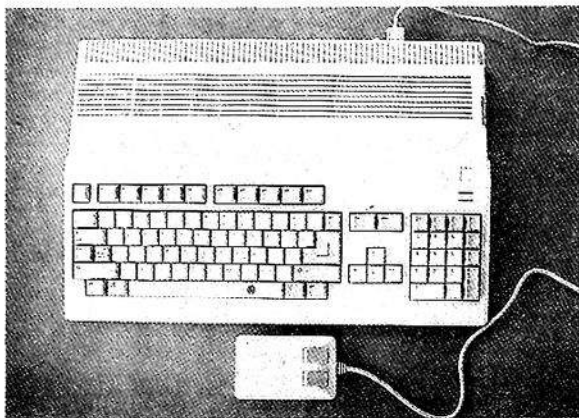
Infatti da Workbench è praticamente impossibile, salvo casi particolari.

Da ambiente Shell (oppure Cli) la cosa è invece *abbastanza* semplice da realizzare, anche se (come sempre) qualche trabocchetto è sempre in agguato. Vediamolo direttamente con un esempio pratico, sfruttando una copia(!) del disco Workbench1.3 originale.

Per il nostro esperimento, si attivi dunque Amiga con la copia di cui sopra e, dopo avere biclickato sull'icona-disco della copia del Workbench, si apra come di consueto la directory **Utilities**.

Prefiggiamoci ora di assegnare l'icona del Notepad al noto programma **More**, già dotato di una sua icona.

Prima di ogni altra cosa, si apra una finestra Shell (doppio click sulla relativa icona) e si provveda ad **eliminare** l'icona di More.

Niente di più facile, visto che questa è rappresentata

da un file dello stesso nome, ma con suffisso "**.info**". Si impartisca dunque, nella finestra Shell, il comando...

```
Cd Utilities
```

...per portarsi nella directory ove sono presenti i file da manipolare e rendere più semplici le successive operazioni. Si digiti ora...

```
Delete More.info
```

...comando che, come vedremo tra breve, potrebbe in realtà essere omesso, ma sarà utile per capire meglio il comportamento generale dell'ambiente Workbench.

Per constatare che l'icona di More non è più presente, si chiuda la finestra della directory Utilities, e la si riapra biclickando nel relativo cassetto: come prevedibile, nella nuova visualizzazione mancherà all'appello il programma More.

In realtà ciò che manca è la sua icona: il file sarà sempre disponibile, come si potrà constatare impartendo un comando **List** nella finestra Shell, o addirittura attivandolo digitando More (e Return) sempre in ambiente Dos (la Shell, per intenderci).

Ma poiché il nostro scopo è quello di mantenerne la visualizzazione da Workbench, seppure con una diversa icona, si digiti ora...

```
Copy Notepad.info More.info
```

Per vedere che cosa è successo, sarà necessario chiudere di nuovo la finestra Utilities e quindi riaprirla.

Apparentemente tutto sembrerà immutato, ma si provi a spostare l'icona Notepad con il mouse: ecco lì il nostro More, dotato però della **stessa icona** di Notepad. La sovrapposizione iniziale, per la cronaca, è dovuta al fatto che il file **.Info** memorizza al suo interno non solo l'immagine grafica, ma **anche** la sua posizione all'interno della directory: essendo **More.Info** una copia esatta di **Notepad.Info**, ecco che le due icone appariranno nella stessa posizione.

Per ovviare, sarà comunque sufficiente clickare una sola volta sull'icona di Notepad dopo averla spostata, e selezionare Snapshot dal menu Special del Workbench.

La procedura appena illustrata, sfoltita da scopi didattici, si può in pratica limitare al solo comando Copy prima descritto, senza la necessità di cancellare l'icona da sostituire: il comando, infatti, sostituirà automaticamente la nuova icona alla vecchia, come farebbe con qualunque altro tipo di file.

Come ovvio, per la sostituzione potrà essere adoperata *qualunque* icona posta in *qualunque* directory, con l'accortezza di specificare l'intero percorso (path) che identifica la posizione dell'icona.

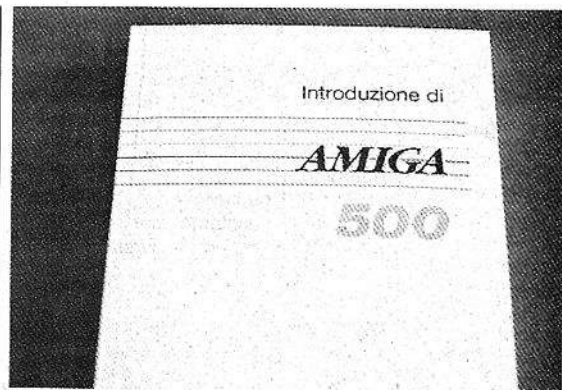
Se, per esempio, avessimo voluto sostituire l'icona di More con quella del programma Diskcopy contenuto nella directory System dello stesso dischetto, avremmo dovuto usare un più verboso comando...

```
Copy
:system/diskcopy.info
More.info
```

...(da digitare su un unico rigo) sempre che ci si trovi ancòra all'interno della directory Utilities.

Una nota importante: quando si effettua la sostituzione di icona, occorre badare sempre all'attributo **Type** della stessa, che deve, come ovvio, risultare uguale a quello originario.

Per accertarsene, è sufficiente clickare una volta su un'icona e selezionare **Info** dal menu **Project** del Workbench.

In alto a sinistra della finestra informativa è descritto il Tipo dell'icona (Tool, Project, Disk, Drawer). Se si prova a leggere la Info delle icone della directory Utilities, si constaterà come queste siano **tutte** di tipo Tool, per cui interscambiabili come si preferisce.

Se, però, si intende assegnare una icona di tipo per esempio **Drawer** (i *cassetti* delle directory) a un programma che ne possiede una di tipo Tool, è indispensabile modificarne prima il tipo, in modo che risulti anch'esso di tipo Tool.

Operazione, questa, effettuabile solo tramite appositi programmi come il **Changer** incluso in **Computer Club Disco n. 1**, oppure adoperando un icon-editor che consenta di caricare l'immagine, e di modificarne l'attributo Tipo *prima* di risalvarla su disco.

A parte queste ultime accortezze, c'è anche da dire che la sostituzione, se non si ha voglia di scomodare il Dos, può essere effettuata adoperando una qualunque Dir Utility (**Disk Master**, **Climate**, eccetera) per copiare il file .Info in **Ram**, assegnargli il nome del nuovo file cui andrà associata l'icona, e quindi copiarlo nella stessa directory del file in questione.

Un compito in definitiva non difficile, ma... neanche troppo facile, se si è alle prime armi. Un buon esercizio, comunque, per una (quasi) lezione su Amiga Dos.

C1-Text e Ms-Dos

Ho letto sulla vostra rivista che con il word processor C1-Text si possono scrivere testi leggibili anche su computer Ms-Dos compatibili, così mi sono affrettato

ad usarlo, visto che a casa ho un Amiga 500 e in ufficio un Pc. Ho usato su C1-Text il set di caratteri Pc Ibm e ho salvato il documento in Ascii. Poi, come da voi consigliato, ho adoperato Dos2Dos per copiare il testo su un disco Ms-Dos. Risultato: sul Pc il testo si leggeva, ma le linee poste dopo un "a capo" erano tutte disallineate. Ho provato a copiare con l'opzione "-a" di Dos2dos (come indicato su CCC n. 84), ma peggio che andar di notte: le righe risultavano allineate, ma le lettere accentate non venivano più riconosciute. E' un problema che si può risolvere?

(Mauro Mursi - Roma)

Certamente, e in modo tra l'altro estremamente semplice.

Il mancato allineamento delle righe, dopo il trasferimento di un testo in ambiente Ms- Dos, è sicuramente da imputare al carattere di Fine Paragrafo, che differisce nei due tipi di computer. Amiga adopera infatti il codice ASCII 10 (Line Feed, abbreviato comunemente con LF), mentre i Pc compatibili adottano due codici, il 13 (Carriage Return, ovvero CR) seguito dal suddetto Lf.

Il problema si pone in questi termini: L'uso con il C1-Text del set di caratteri Ibm Pc non comprende una trasformazione del codice di fine paragrafo.

Per ottenere una completa emulazione dell'ASCII implementato negli Ms-Dos è necessario un ulteriore settaggio, ricorrendo all'opzione **Parametri** del menu **Formato File**, che comprende come prima voce proprio il tipo di Fine Riga / Fine Paragrafo da adottare. Basterà scegliere (con il mouse) **Cr+Lf**, e tutto sarà... quasi risolto.

Il "quasi", però, non riguarda direttamente C1-Text, quanto piuttosto la modalità scelta per copiare il file Ascii elaborato. Adoperando Dos2Dos, le alternative sono in pratica due: se il testo comprende lettere accentate dell'alfabeto italiano, allora è indispensabile il settaggio del Fine Paragrafo come appena detto, e la copia su floppy Ms-Dos andrà eseguita **senza** usare l'opzione **"-a"** di Dos2Dos.

Se nel documento non sono invece adoperati caratteri speciali (come appunto le vocali accentate), allora non si renderà necessario modificare il normale Fine Linea (/paragrafo) di Amiga, che sarà automaticamente tradotto dal Dos2Dos adoperato congiuntamente al parametro "-a".

Amiga 500 regge?

L'Amiga 500 riesce a sopportare contemporaneamente gli accessori A501 e A590 dal punto di vista dell'alimentazione elettrica?

(Una voce dal profondo Sud)

Il problema non sussiste. Intanto l'hard disk A590 è dotato di un suo alimentatore esterno.

Come se non bastasse, il suo eventuale acquisto rende **inutile** l'espansione di memoria A501, in quanto è possibile aggiungere comodamente fino a **due** Megabyte di ram nella scheda controller contenuta al suo interno, che verranno *visti* da Amiga come normale memoria (fast) di espansione.

Da computer a videotape

Dopo innumerevoli sforzi, sono finalmente riuscito a registrare delle schermate fatte col Dpaint su una videocassetta utilizzando il modulatore Rf. Il risultato però è deludente: scarsa chiarezza dei colori e fastidioso tremolio dell'immagine.

Questi effetti scompaiono se si adopera qualcosa come il videogenlock da voi recensito sul n.85? Inoltre: sovrapponendo le immagini del computer su quelle video non si perderà per caso l'audio originale della cassetta?

(E. Gnasso - C.le Monferrato)

L'uso di genlock amatoriali può senza dubbio migliorare la qualità di riproduzione di immagini provenienti dal computer rispetto all'uso di un modulatore, per di più consentendo la sovrapposizione di un altro segnale proveniente dall'esterno.

Non ci si aspettino comunque miracoli, se si adoperano genlock amatoriali che sfruttano il segnale videocomposito (non quello Rgb) di Amiga e della fonte esterna.

Come più volte ribadito nelle recensioni che li riguardano, i genlock di questo livello sono un buon compromesso tra costo e prestazione, ma non possono certo essere paragonati ad altri prodotti di ben più alte prestazioni e dal prezzo che normalmente supera abbondantemente il milione e mezzo.

Quanto al secondo dubbio, non ha motivo di esistere: il segnale video **proviene** da una eventuale videocassetta fornita di sonoro, viene miscelato al segnale di Amiga **all'interno** del genlock, e l'output

va poi diretto ad un altro videotape e relativa cassetta, che nulla ha da spartire con quella originale.

L'audio, comunque, non viene trasferito affatto attraverso il genlock o le sue connessioni.

Se lo si desidera, sarà necessario effettuare separatamente i collegamenti necessari a registrarlo, prelevandolo dal tape sorgente o dal computer, a seconda di ciò che si intende realizzare.

Nostalgico

Ho appena cambiato il mio vecchio C/64 con un Amiga 500. Prima adoperavo la user port con una serie di Poke per gestire interfacce hardware molto utili. Ora vorrei sapere: possibile che non esista un modo per rendere compatibili le schede per le due macchine Commodore?

(Raffaele Pesarini)

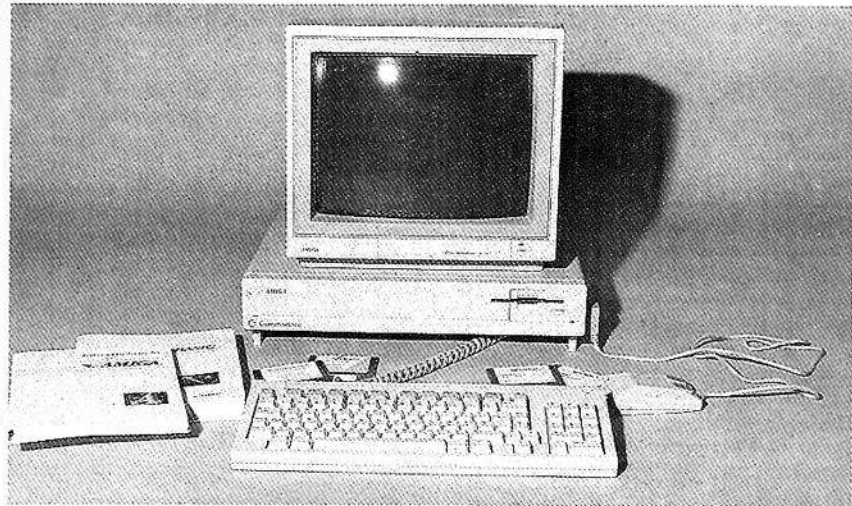
No, non esiste modo. I due computer sono diversi anni luce, meglio rassegnarsi e cominciare ad attrezzare il nuovo gioiello.

Resuscitazioni

Esiste un programma in grado di catturare le immagini rimaste in memoria dopo un reset? Si può fare la stessa cosa con la musica?

(A. Buonanni - Rivara C.se)

Esistono entrambi i tipi di programma. Per ciò che riguarda le immagini, il più noto è forse **Third Day**, inserito anche in uno dei nostri dischi Amigazzetta (il n. 8).

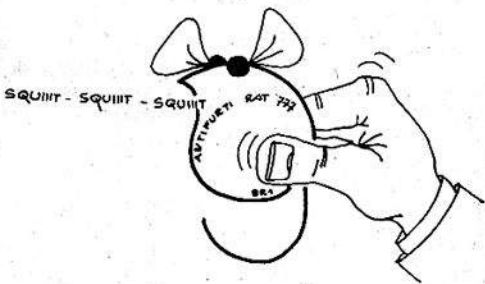
Il risultato non è sempre garantito, molto dipende dal tipo di codifica adottato dalla grafica utilizzata da un programma, ma è l'unica alternativa software alle ben più potenti risorse di cartucce come la **Action Replay**. Anche per le musiche vale lo stesso discorso, compresa la superiorità di un mezzo hardware come la Action Replay, che spazia agilmente anche in campo sonoro.

Esiste comunque una vasta disponibilità di software in grado di recuperare anche dopo un reset il contenuto di moduli musicali per lo più in formato **Sound Tracker**, con vari nomi quasi sempre comprendenti un termine *ripper* (per esempio **Sound Ripper**).

Per la cronaca, **Amigazzetta 9** contiene uno di questi programmi, di nome **Stu**.

C'è Ram e Ram

Che differenza c'è tra la Chip memory e la Fast memory?

(Alessandro Consoli)

La differenza consiste nell'accesso che vi hanno gli elementi più importanti di Amiga: il microprocessore centrale (Cpu) e i cosiddetti Chip Custom, preposti a disbrigare molti compiti che, in caso contario, rallenterebbero, e di molto, le numerose attività del 68000 e di Amiga.

In pratica: alla memoria **Chip** accedono tanto la Cpu che i chip Custom, mentre alla cosiddetta **Fast** può accedere **solo** il microprocessore centrale.

Il termine Fast, la cui traduzione dall'inglese è *veloce*, sta proprio a indicare che, non dovendo condividere gli accessi con nessun altro, il 68000 vi opera con maggiore velocità. Molti dati, di contro, richiedono che i programmatori si preoccupino di collocarli in Chip memory (grafica, suono, buffer di dischi, eccetera), **unico** luogo ove possono essere gestiti adeguatamente. La Chip memory assomma, nei modelli Amiga di *recente produzione*, **1 Mega byte** di ram, mentre in quelli immediatamente precedenti non superava i 512 Kb. Tutta la memoria eccedente queste dimensioni, montata sotto forma di espansioni, viene considerata Fast ram.

Un'eccezione, a onor di cronaca, è rappresentata dagli **Amiga 3000**, il cui Agnus (si chiama così il chip che indirizza la chip memory) può gestire fino a 2 Megabyte di Chip.

✔ Amiga	❑ Principianti	✔ Esperti	❑ Tutti	✔ HELP
✔ Ms - Dos				
❑ Recensioni				
❑ Hardware				
❑ Software				
❑ Applicazioni				
✔ Programmazione				
❑ Dos				
✔ Pascal				
❑ C				
❑ Basic				
❑ Assembly				

Vediamo quali istruzioni attivare per generare musica utilizzando il noto compilatore Kick Pascal per Amiga.

Musica in Pascal

< di Filippo Bosi >

< Esaminare C. C. n. 85, n. 86, n. 87 >

Interrompendo per motivi di spazio la stesura della unit **TPGrafica** (di cui ci siamo occupati fino al numero scorso), vediamo stavolta come generare suoni con **Kick Pascal**, avendo sempre un occhio di riguardo verso il mondo PC (o, meglio, verso il **Turbo Pascal**). Premessa fondamentale, valida non solo per Kick Pascal, ma per *qualsiasi* linguaggio di programmazione: esistono due modi di generare suoni con Amiga, il primo **accedendo direttamente all'hardware**, il secondo **tramite il device audio**, una interfaccia software fornitaci da AmigaDos.

Purtroppo, contrariamente a quanto si potrebbe pensare in un primo momento, spesso e volentieri la seconda strada si rivela più complicata della prima, in quanto deve rispettare le ferree regole del **multitasking**: l'hardware di Amiga mette infatti a disposizione "solo" quattro canali audio, mentre il software permette di fare girare contemporaneamente più programmi, che magari pretendono lo stesso canale...

Ecco allora che, ogniqualvolta si voglia suonare anche una piccola, unica nota, siamo costretti a scrivere svariate linee di codice, con chiamate a routine di I/O verso il device audio. Saremo comunque sicuri che il nostro programma funzionerà tranquillamente insieme ad altri, altrettanto corretti nell'accedere alle risorse di Amiga.

Al fine di comprendere meglio il contenuto dell'articolo di questo mese, riteniamo necessarie due parole sulla sintesi sonora computerizzata.

Uno strumento elettronico, per generare suoni, crea *oscillazioni elettroniche*, all'interno dei suoi circuiti, corrispondenti al suono richiesto: non possiamo udire tali oscillazioni fino a quando non vengono convertite in *oscillazioni sonore* (compressioni di masse d'aria) da altoparlanti che, nel nostro caso, sono quelli del monitor. Quali parametri determinano il tipo di suono che esce dal computer?

Generazione di effetti sonori con Amiga

Il range di valori consentito per ogni **sample** (= campione) varia tra -128 e +127: tale intervallo dovrebbe essere utilizzato nel maggior modo possibile.

Se, ad esempio, dobbiamo costruire un'onda quadra, **non** ha senso utilizzare la serie di valori (-10, 10), quando risulta molto più forte, a parità di volume, un suono generato da (-128, 127).

Se si deve controllare il volume di un suono, si agisca sul parametro del **volume** e **non** sull'ampiezza delle oscillazioni dell'onda.

Ogni ciclo di digitalizzazione dovrebbe iniziare con **0** e terminare con **0**, per evitare ticchettii quando si cambia forma d'onda. Desiderando suonare una nota di una certa frequenza, dobbiamo fornire all'**audio.device** il valore del periodo, calcolato secondo una formula, simile a quella presente all'interno della routine PlaySample.

Per motivi legati all'hardware, il valore del periodo può variare da 124 a circa 500.

Per una digitalizzazione di 32 byte si possono quindi ottenere solamente frequenze da **260 Hz** a **523 Hz**.

Per spaziare oltre questi limiti bisogna raddoppiare o dimezzare la risoluzione del campionamento.

Più lunghi sono i dati, minore sarà la frequenza massima possibilei.

Viceversa, più corti saranno i dati, più frequenze alte possiamo raggiungere.

Prima di tutto la **frequenza**, che determina l'altezza di un suono, misurata in Hertz (Hz), corrispondente al numero di oscillazioni in un secondo.

Secondo parametro è il **volume**, inteso come ampiezza delle oscillazioni.

Terzo parametro, infine, è il **timbro**, molto più complicato dei primi due, ma sicuramente il più importante: esistono infatti tantissimi strumenti musicali che possono suonare con la stessa frequenza e lo stesso volume, ma hanno timbro differente: il timbro è dato dalla forma d'onda dell'oscillazione.

Proprio la possibilità di definire in modo semplice ed efficace la forma d'onda è uno dei principali vantaggi che Amiga possiede nel campo della sintesi sonora rispetto ad un computer MS - DOS privo di scheda musicale.

In quest'ultimo, infatti, l'unico parametro che si può controllare via software è la frequenza del suono generato: il volume è controllabile solo via hardware, e non in tutti i modelli; il timbro è poi impossibile da definire, a meno di ricorsi a trucchetti in assembler che portano, in ogni caso, a risultati che lasciano alquanto a desiderare.

Veniamo ora al dunque descrivendo la logica di creazione di un suono con Amiga; prima di tutto si deve chiedere al device audio di riservare per il nostro programma un canale audio tra quelli attualmente liberi; dopodichè dobbiamo fornire ad Amiga la descrizione del suono da creare.

Per quanto riguarda volume e frequenza, essi consistono in due valori numerici, quindi la loro rappresentazione all'interno di un computer non presenta grossi problemi; ma per quanto riguarda la forma d'onda? Si ricorre ad una digitalizzazione, ovvero alla "descrizione" di qualcosa di *continuo* tramite una serie di numeri finiti.

Supponiamo di disegnare la forma d'onda desiderata su di un piano cartesiano: suddividendo l'asse **x** in parti uguali otterremo, per ogni divisione, un valore di ampiezza dell'onda sull'asse **y**; avremo così una serie di valori numerici, facilmente rappresentabili all'interno del computer, che descrivono la forma d'onda più o meno fedelmente a seconda del numero di suddivisioni (**frequenza di campionamento**).

Dopo aver digitalizzato la forma d'onda desiderata ed averla trascritta in memoria, dobbiamo comunicare ad Amiga di convertirla in onde sonore: con una chiamata all'**audio.device** otteniamo il suono desiderato. Per fare terminare il suono occorrerà un'altra chiamata al device audio, tramite comando opportuno.

Il procedimento, di facile descrizione, purtroppo non ha realizzazione immediata. Una volta, però, realizzate le routine di output sonoro, ci troviamo di fronte a potenzialità incredibili: molto spesso programmare Amiga richiede un piccolo sforzo iniziale, ma i risultati lo ripagano sicuramente.

Come compilare il programma in Kick Pascal

1 - Digitare all'interno dell'editor KP il codice sorgente di **FormeOnda.p**, ricordandosi di memorizzare il codice sorgente su disco... Non si sa mai!

2 - Compilare con il comando **C** se in modo linea, usando i menu oppure **Shift + F9**. Se tutto è andato bene, otteniamo i messaggi informativi del linker sulla lunghezza del programma e sulla suddivisione in Hunks.

Se si desidera avere il codice eseguibile su disco, allora....

3 - dopo avere compilato il programma (punto 2), battere in modo linea **E**: verrà chiesto il nome del file su cui registrare l'eseguibile.

Se supponiamo di rispondere `ram:FormeOnda`, troveremo, nella ram disk, il programma `FormeOnda`, eseguibile da CLI, direttamente battendo FormeOnda + Return.

I programmi compilati con Kick Pascal, che prevedono output a video sulla finestra CLI, **non** possono essere eseguiti da WorkBench. Vedremo prossimamente come aggirare questa limitazione impostaci non da KP ma da AmigaDOS.

Il listato

Detto ciò analizziamo il programma pubblicato, che può essere diviso logicamente in tre parti: **inizializzazione** di forme d'onda, **interfacciamento** con il device audio e parte **dimostrativa**.

Le prime due parti possono facilmente essere utilizzate come base per la costruzione di programmi molto più complessi, mentre l'ultima vuole essere un piccolo esempio di utilizzo delle routine.

Le routine di inizializzazione di forme d'onda (**SetupSquareSample**, **SetupSineSample**, **SetupNoiseSample**) non fanno altro che allocare un'area di memoria chip (notare **MEMF_CHIP** in **Mem_Alloc()**), grande abbastanza da contenere la descrizione della forma d'onda, espressa come array di Short (Interi a 8 bit con segno, cioè da -127 a +127).

Si osservi come buona pratica di programmazione nella costruzione di forme d'onda sia quella di avere il valore **0** nel primo e nell'ultimo *sample* di ogni forma d'onda, per limitare eccessivi stacchi tra una nota e l'altra.

A complemento di queste routine è presente la **FreeSample**, necessaria ogniqualvolta si richieda il cambiamento di forma d'onda. Essa libera l'area di memoria riservata alle routine precedenti.

La parte di interfacciamento con il device Audio è costituita dalle procedure **InitAudio**, **PlaySample** e **NoSound**. Queste ultime due non fanno altro che impartire comandi al device e, rispettivamente, emettere una forma d'onda con una data frequenza o terminare la produzione di suoni.

La prima, invece, apre il device (**Open_Device**) audio, *dopo* avere creato una porta di comunicazione con lo stesso.

Degna di nota, all'interno di quest'ultima procedura, è la definizione della struttura **AllocTable[]**: l'array, costituito di costanti binarie a quattro bit (**%xxxx**), corrispondenti ognuno ad un canale audio di Amiga, indica le varie configurazioni di canali che il programma può accettare dal device audio.

Segue subito la richiesta di allocazione, che punta ad AllocTable, inviata con un banale **BeginIO()**.

In questo caso abbiamo bisogno di uno qualsiasi dei 4 canali di Amiga, non ha importanza se destro o sinistro, quindi specifichiamo le quattro configurazioni possibili di un unico canale.

Ottenuto il canale libero, lo blocchiamo chiedendo un **lock** (=lucchetto). Dovremo però verificare che la richiesta del lock sia andata a buon fine, poichè potrebbe capitare che, nell'intervallo di tempo passato dalla richiesta dei canali disponibili alla richiesta del lock, un altro programma abbia "rubato" il canale, segnalato come *libero* al nostro programma (potenza del multitasking! si deve stare attenti non solo a quello che fa il nostro programma, ma anche a quello che possono fare gli altri!). Non è escluso, comunque, che, nonostante gli accorgimenti usati, il programma compilato, una volta fatto partire, riesca a inchiodare Amiga, pur senza emettere segnalazioni di errore o Guru.

Consigliamo, quindi, di attivare il programma solo dopo aver resettato Amiga o dopo essersi assicurati, quanto meno, che nessun programma sia contemporaneamente in azione.

Analizziamo infine, molto brevemente, la parte dimostrativa del programma pubblicato: essa è costituita dalla routine **PlayScale** e dalla porzione di programma principale.

La prima si limita a chiamare la routine **PlaySample** con valori di frequenza scelti con un ciclo **for**, mentre la seconda esegue la PlayScale variando la forma d'onda del suono prodotto mediante chiamate alle **SetupSample**.

Effetti sonori in Turbo Pascal

TP mette a disposizione due routine per la creazione di suoni:

`Sound (<frequenza>)`

...e...

`Nosound`

La seconda trova perfetto equivalente nella routine **NoSound** presente nel listato di questo mese, mentre la prima può essere facilmente emulata da una chiamata alla routine **PlaySample**, sempre appartenente al programma che appare in queste pagine.

Ciò può verificarsi, ovviamente, **dopo** avere chiamato **InitAudio** ed avere definito una forma d'onda in memoria, generalmente quadra (vedi programma).

Particolare attenzione bisogna prestare alle frequenze da generare, proprio per il fatto che queste dipendono dalla lunghezza, in byte, del sample come descritto nel riquadro specifico. Si dovrà quindi generare una (o più) serie di valori in numero tale da generare tutte le frequenze richieste, senza eccedere i limiti (124 ... 500) della variabile **Rate** (periodo) della routine PlaySample.

Spesso, nei programmi musicali, vengono inseriti dei **ritardi**.

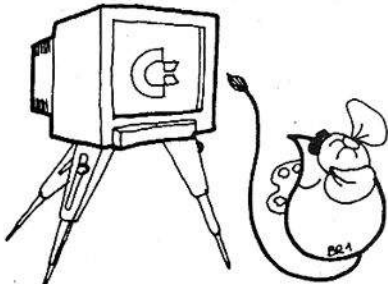
In **TP** si utilizza la routine...

`Delay (<millisecondi>)`

...che arresta il flusso del programma per un numero specificato di **millisecondi**. Esiste anche una routine **Delay()** in Amiga, ma richiede un ritardo espresso in **ticks** (corrispondenti alla frequenza del quadro video, cioè 1/50 di secondo). Si deve ricorrere quindi alla seguente equivalenza: Delay(1000), in **TP**, equivale a Delay(50) in **KP**. Avremo quindi la possibilità di specificare in KP il minimo ritardo con **Delay(1)**, corrispondente a **Delay(20)** in TP. In altre parole: Delay(x) in TP equivale a Delay(x/20) in KP.

```
Program FormeOnda;
{ Programma dimostrativo per l'utilizzo }
{ di audio.device - by Filippo Bosi     }
{ per Computer Club - OTTOBRE 91        }

Uses ExecSupport, ExecIO;

{$incl 'devices/audio.h',
 "workbench/startup.h",
 'exec/memory.h' }

Const
  CLOCK = 3579545;

Type
 SamplePtr=^array [0..MaxLongInt]
           of short;

Var
  MySample            : SamplePtr;
  LunghSample         : Long;
  allocIOB, lockIOB   : ^IOAudio;
  port                : ^MsgPort;
  mydevice            : p_Device;
  err                 : Long;
  Suono               : boolean;

procedure SetupSquareSample;
{ crea in memoria una forma d'onda }
{ quadra                           }
var sp: SamplePtr;

Begin
 LunghSample:=4;
 sp := Ptr (Alloc_Mem (LunghSample,
                       MEMF_CHIP));
 MySample:=sp;
 MySample^[0]:=0;
 MySample^[1]:=127;
 MySample^[2]:=-127;
 MySample^[3]:=0;
End;

procedure SetupSineSample;
```

```
{ crea in memoria una forma d'onda }
{ sinusoidale                      }
var sp: SamplePtr;
    i:  LongInt;
    sample: Real;
Begin
LunghSample:=20;
sp:=Ptr(Alloc_Mem(LunghSample,MEMF_CHIP));
MySample:=sp;

for i:=0 to (LunghSample-1) do
  begin
  sample:=127*sin(2*pi*i/(LunghSample-1));
  MySample^[i]:=round(sample);
  End;
end;

procedure SetupNoiseSample;
{ crea in memoria una forma d'onda }
{ casuale, che genera un suono     }
{ distorto                         }

 var i:byte;
     sp: SamplePtr;

 Begin
 Randomize;

 LunghSample:=70;
 sp:=Ptr(Alloc_Mem(LunghSample,
                   MEMF_CHIP));
 MySample:=sp;

 for i:=0 to (LunghSample-1) do
    begin
    MySample^[i]:=random(250)-125;
    End;
 End;


procedure FreeSample;
{ libera la memoria occupata da una }
{ forma d'onda                      }

  Begin
    Free_Mem(long(MySample),LunghSample);
  End;


Procedure Fatal_Error;
{ rientra chiudendo tutte le risorse }
{ allocate, in caso di errore fatale }
begin
 if (port<>Nil) then DeletePort(port);
 if (allocIOB<>Nil) then
   begin
   Close_Device(allocIOB);
   DeleteExtIO(allocIOB);
   end;
 if (lockIOB<>Nil) then
   DeleteExtIO(lockIOB);
end;
Procedure InitAudio;
{ apre Audio.device, il Port associato }
{ e riserva un canale audio....        }

Var alloctable : Array[1..4] Of Byte;
Begin

port :=Nil;
port := CreatePort ('KPsoundPORT', 0);
If port=Nil Then Fatal_Error;


allocIOB:=Nil;
allocIOB:=CreateExtIO(port,
                      SizeOf(IOAudio));
If allocIOB=Nil Then Fatal_Error;

lockIOB:=Nil;
lockIOB:=CreateExtIO(port,
                      SizeOf(IOAudio));
If lockIOB=Nil Then Fatal_Error;

Open_Device(AUDIONAME, 0, AllocIOB, 0);

mydevice:=allocIOB^.ioa_Request.io_Device;
lockIOB^.ioa_Request.io_Device:=mydevice;

AllocTable[1] := %0001;
AllocTable[2] := %0010;
AllocTable[3] := %0100;
AllocTable[4] := %1000;

With allocIOB^, ioa_Request, io_Message Do
  Begin
  io_Flags := ADIOF_NOWAIT;
  ioa_Data := ^AllocTable;
  ioa_Length := 4;
  io_Command := ADCMD_ALLOCATE;
  BeginIO(allocIOB);
  End;

err := WaitIO(allocIOB);
If err <> 0 Then
    begin
    Writeln('Allocazione non riuscita');
    Fatal_Error;
    end;

With lockIOB^, ioa_Request Do
  Begin
  io_Unit:=allocIOB^.ioa_Request.io_Unit;
  io_Command:=ADCMD_LOCK;
```

```
   ioa_AllocKey:=allocIOB^.ioa_AllocKey;
   End;
 SendIO(lockIOB);
 If CheckIO(lockIOB) <> 0 Then
       Begin
       Writeln('Canale Rubato!');
       Fatal_Error;
       end;

 Suono:=FALSE;

 End;


 Procedure NoSound;
 { termina l'emissione di eventuali }
 { suoni dal canale audio allocato  }

   Begin
     if (Suono=TRUE) then
        begin
        With lockIOB^, ioa_Request Do
           Begin
           io_command:=ADCMD_FINISH;
           io_Flags:=IOF_QUICK;
           End;
        BeginIO(lockIOB);
        Err:=WaitIO(lockIOB);
        Suono:=FALSE;
        end;
   End;


Procedure PlaySample(s: SamplePtr;
                     Freq:LongInt);
{ comanda all'audio.device di suonare }
{ la forma d'onda puntata da s con    }
{ frequenza <Freq>                    }
{ OSSERVAZIONE: limitazione su <Freq> }
{  <Freq> deve essere tale che rate   }
{  sia compresa tra 124 e 500(ca),    }

Var Rate: Long;

Begin
 if (Freq<=0) then exit;
 Rate:=CLOCK div (LunghSample * Freq);
 if (Rate<124) then Rate:=124;

 NoSound;

 With lockIOB^, ioa_Request Do
   Begin
    io_Command := CMD_WRITE;
    io_Flags := ADIOF_PERVOL + IOF_QUICK;
    ioa_Data := ^s^;
    ioa_Length := LunghSample;
    ioa_Volume := 64;
    ioa_Period := Rate;
    ioa_Cycles := 0;
   End;
BeginIO(lockIOB);
 Suono:=TRUE;
End;

Procedure PlayScale;
{ suona alcune note in scala }

var i:integer;

begin
  for i:=1 to 10 do begin
        PlaySample(MySample,i*40);
        Delay(10);
        end;
end;

{ ** programma principale ** }

Begin
Writeln('Suoni in KICKPASCAL - by FB');
Writeln('---------------------------');

Writeln('Apro un canale audio...');
 InitAudio;

Writeln('Forma d''onda quadra');
  SetupSquareSample;
  PlayScale;
  NoSound;
  FreeSample;

Writeln('Forma d''onda sinusoidale');
  SetupSineSample;
  PlayScale;
  NoSound;
  FreeSample;

Writeln('Forma d''onda casuale (rumore)');
  SetupNoiseSample;
  PlayScale;
  NoSound;
  FreeSample;

{* esci dal programma e libera *}
{* le risorse allocate...      *}

Close_Device(allocIOB);
DeletePort(port);
DeleteExtIO(allocIOB);
DeleteExtIO(lockIOB);
End.
```

✔ Amiga	❑ Principianti	❑ Esperti	✔ Tutti	✔ HELP
✔ Ms - Dos				Impariamo a "programmare" la nostra stampante con la massima facilità possibile.
❑ Recensioni				
❑ Hardware				
❑ Software				
✔ Applicazioni				
❑ Programmazione				
❑ Dos				
❑ Pascal				
❑ C				
❑ Basic				
❑ Assembly				

Come stampare su più colonne

< di Ascanio Orlandini > — < Esaminare C. C. n. 87 >

Abbiamo sottolineato, sul n. **87**, come le stampanti attualmente commercializzate siano dotate di elaborate funzioni che raramente vengono sfruttate.

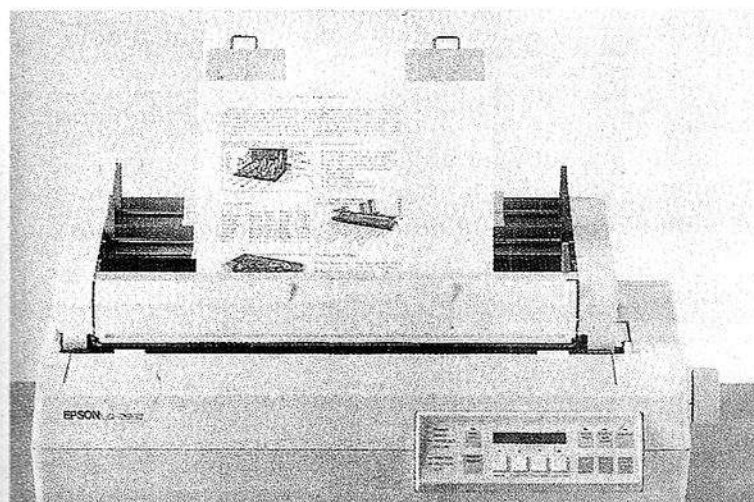
Molti documenti che leggiamo abitualmente hanno una *formattazione* a colonne (quotidiani, riviste, libri, ecc.).

Un simile risultato è ottenibile mediante un'accurata e complessa elaborazione del documento presente in memoria RAM (e successiva stampa per righe orizzontali), oppure è necessario stampare una colonna alla volta con la conseguente necessità di fare "passare", la medesima pagina, tante volte quante sono le colonne e riposizionando ogni volta, naturalmente, il foglio di carta in testa al rullo della stampante.

Con il programma di queste pagine riusciremo ad effettuare una stampa su **n** colonne adottando il secondo metodo, ma facendo compiere automaticamente tutte le formattazioni e le giustificazioni del caso alla stampante.

Il funzionamento è garantito per le stampanti **Epson LQ-1000** e **IBM GraphicPrinter** o compatibili. Mediante due semplici modifiche sarà possibile fare funzionare il programma anche su **IBM ProPrinter**.

Il linguaggio adottato è il **Gw-Basic 3.30**, per il mondo MS - DOS, e l'**AmigaBasic 1.2** per Amiga. Non bisogna stupirsi se il listato proposto è unico: essendo entrambi gli interpreti Basic realizzati dalla stessa Microsoft, sono perfettamente compatibili, pur se il formato del listato è diverso da quelli cui sono abituati gli utenti Amiga: AmigaBasic è, infatti, in grado di gestire listati corredati da numeri di linea, e salti come *Goto* e *Gosub*.

Essendo i listati puramente dimostrativi, viene lasciato al lettore ogni genere di abbellimento e perfezionamento.

Come funziona

Il funzionamento del programma è estremamente semplice in quanto si basa sulle capacità di impaginazione della stampante:

⇨ Determina le dimensioni delle colonne.

⇨ Calcola il valore dei margini da usare di volta in volta per ciascuna di esse.

⇨ Apre il file di testo da stampare.

⇨ Invia alla stampante i margini ed il comando di giustificazione bilaterale.

⇨ Invia il testo da stampare.

⇨ Tiene un conteggio approssimativo delle linee stampate e...

⇨ ... dopo aver stampato la prima colonna **invia un codice di FormFeed "al contrario" con l'effetto, quindi, di riavvolgere la pagina all'inizio!**

⇨ Vengono quindi re-inviati i nuovi margini e viene stampata una nuova colonna; e così via fino all'ultima colonna desiderata.

⇨ Raggiunta la fine dell'ultima colonna, viene inviato un normale FormFeed che porta la testina di stampa sulla nuova pagina ed il processo continua fino alla fine del testo da stampare.

Tecnicamente è necessario aprire **due** canali di Input / Output: quello relativo al testo ASCII da stampare *(in lettura)* e quello per la stampante *(in scrittura)*, per leggere il testo nel primo e scriverlo nell'altro (diretto verso la stampante).

In Basic tali operazioni si effettuano con i comandi **Open**, **Input #n** e **Print #n**.

Con **Open** si apre un file (o device) in lettura e/o scrittura, associando un numero che, in seguito, viene usato da **Print #n** e **Input #n** per riferirsi univocamente al particolare canale di Input / Output.

Per maggiori dettagli sui comandi Basic riferitevi al manuale ed esaminate il listato: un esempio ben compreso è sempre più significativo di qualunque discorso.

I codici per la stampante vengono inviati sfruttando i codici decimali mediante la funzione Basic **Chr$ (nn)**, che restituisce il carattere ASCII del codice **nn**, come indicato nella tabella dei comandi usati e nell'esempio.

Compatibilità

Il listato e la tabella fanno riferimento alle stampanti **Epson LQ-1000**, **IBM_GraphicPrinter** e **IBM_ProPrinter**, oltre naturalmente a **tutte** quelle dichiarate compatibili con una delle tre elencate.

Con stampanti diverse, o qualora non venissero riconosciuti alcuni codici usati, si faccia riferimento al manuale della stampante posseduta, cercando nell'indice analitico le stringhe suggerite nella tabella stessa alla voce, appunto, *Voce indice analitico.*

Analizzando il listato

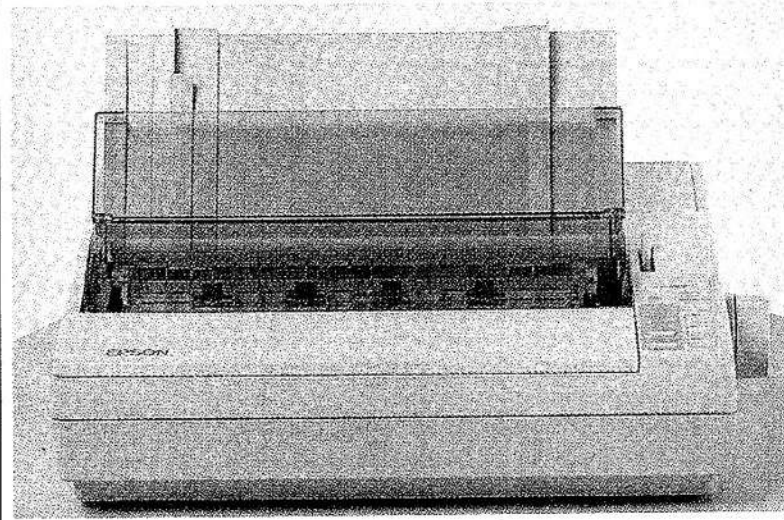
Righe 20-50: i dati vengono inseriti direttamente nelle variabili. **NRcolonne** contiene il numero di colonne nelle quali verrà stampato il documento (mantenere valori da **2** a **5**); **InterColonna** è il numero di spazi che separeranno le varie colonne tra di loro; **MargSin** e **MargDest** sono gli spazi lasciati, rispettivamente, a sinistra della prima colonna e a destra dell'ultima; **NRlinee** contiene invece il numero di linee della pagina calcolate approssimativamente per *difetto.* Si sottolinea che quest'ultimo dato deve essere inferiore di almeno **10** al numero effettivo di linee stampabili, in quanto il calcolo delle righe stampate è approssimativo dal momento che la stampante inserisce spazi autonomamente per giustificare il testo, con il risultato di allungare quindi il documento stampato ad insaputa del programma.

Righe 65-75: in queste righe ci si preoccupa di aprire, **in lettura**, il file testo da stampare ed **in scrittura** il canale della stampante. Gli utenti Amiga dovranno stare attenti a modificare la linea 70. **LPT1** andrà infatti sostituito da **PAR:** e non da **PRT:** in quanto quest'ultimo invia dati alla stampante *filtrandoli* attraverso il **printer.device** ed il driver stampante selezionato. Il printer.device, ed il driver, hanno il compito di *trasformare* i dati ricevuti in "qualcosa" di stampabile e quindi, ricevendo dati binari anziché caratteri ASCII (il codice di ESC è binario...) tenterà di fare il suo dovere interpretandolo in qualche modo.

Attraverso il device PAR:, invece, si ha **diretto** accesso al **parallel.device** (cioè la porta parallela), *qualunque* periferica vi sia collegata. Si ha quindi la certezza che tutti i dati inviati, binari oppure ASCII, arriveranno a destinazione senza modifiche o false interpretazioni da parte del sistema.

Righe 90 - 95: viene calcolata e comunicata la dimensione della colonna (variabile **Colonna**).

Riga 110: viene mandata alla stampante la sequenza di **Reset**, cioè la si riporta nelle condizioni immediatamente successive all'accensione.

Riga 115: si attiva la formattazione di stampa giustificata ed allineata da entrambi i lati.

Voce indice analitico: **Reset code**
Scopo: reinizializzare la stampante ai parametri di default.
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	ESC	"@"
(decimale)	27	64
(esadecimale)	1B	40

Voce indice analitico: Allineamenti - **Aligning Text**
Scopo: Selezionare allineamenti, giustificazioni o centramenti
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	Esc	"a"	n
(decim.)	27	97	n
(esad.)	1B	61	n

Note: n può assumere valori di 0, 1, 2 oppure 3 per ottenere, rispettivamente, il testo allineato a sinistra, centrato, allineato a destra o allineato a destra e sinistra (cioè giustificato).

Voce indice analitico: Margini - **Margins**
Scopo: impostare il margine destro sulla stampante
Compatibilità: Epson LQ-1000, IBM-G

Codici:	Esc	"Q"	n
(dec.)	27	81	n
(esad.)	1B	51	n

Compatibilità: IBM-P

Codici:	<FS>	"Q"	n
(dec.)	28	81	n
(esad.)	1C	51	n

NOTE: Il valore di **n** deve essere compreso tra 2 e 255 e viene ignorato se non supera il margine sinistro.

Voce indice analitico: Margini - **Margins**
Scopo: impostare il margine sinistro sulla stampante
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	Esc	"l"	n
(dec.)	27	108	n
(esad.)	1B	6C	n

Note: il valore di **n** può variare da 0 a 255 e viene ignorato se supera il margine destro.

Voce indice analitico: Margini - **Margins**
Scopo: impostare contemporaneamente il margine sinistro e destro
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	Esc	"X"	n1	n2
(dec.)	27	88	n1	n2
(esad.)	1B	58	n1	n2

Note: n1 può variare da 0 a 255; n1 da 1 a 255. Se i margini si sovrappongono vengono ignorati (attenzione: prima viene impostato il margine sinistro che, se viene posto *oltre* al margine destro preesistente, viene ignorato!)

Voce ind. an.: Form Feed inverso - **Form Feed reverse**
Scopo: riavvolgere il foglio di carta all'inizio della pagina corrente.
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	Esc	<FF>
(dec.)	27	12
(esad.)	1B	0C

Nota: il comando è ignorato quando è installato l'inseritore dei fogli singoli.

Voce indice analitico: Form Feed - FF - **<FF>**
Scopo: avanzare la carta fino alla pagina successiva
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	<FF>
(dec.)	12
(esad.)	0C

Nota: funziona correttamente solo se è impostata la corretta lunghezza della carta.

Voce ind. an.: Campanello, Suoneria, Bell, Ring, **<BEL>**
Scopo: fare suonare il "campanello" della stampante per 1/4 di secondo
Compatibilità: Epson LQ-1000, IBM-G, IBM-P

Codici:	<BEL>
(dec.)	7
(esad.)	07

Tabella dei codici usati nel listato pubblicato.

Riga 130: viene posto a **1** il numero di pagina corrente (variabile **Pag**).

Loop 135 - 255:
corpo principale del programma che si ripete finché sono state stampate tutte le colonne inserite in NRcolonne.

Righe 145 - 155: vengono calcolati i margini sinistro e destro della colonna corrente e comunicati insieme al numero di pagina.

Riga 165: viene impostato, sulla stampante, il margine destro. I possessori di stampanti IBM ProPrinter devono sostituire **Chr$ (27)** con **Chr$ (28)**.

Riga 175: viene impostato il margine sinistro.

Riga 180: viene impostato nuovamente il margine destro. Questa operazione ripetuta si rende necessaria in quanto la stampante non accetta margini che si sovrappongono. Spieghiamo meglio con un esempio: è valida una impaginazione con margine sinistro di 57 e destro di 66, mentre ne viene ignorata una con sinistro a 58 e destro a 43, che in effetti non avrebbe senso. Supponiamo che si stia stampando l'ultima colonna con margini 60 (sinistro) e 75 (destro). Arrivati in fondo, si passa alla pagina nuova stampando a partire dalla prima colonna; si devono impostare quindi i margini a 5 (sinistro) e 20 (destro). Il programma dapprima tenta di settare il margine destro a 20 (riga 165), settaggio che viene ignorato

```
10 REM Programma per stampare su piu' colonne
15 REM Versione unica AmigaBasic, Gw-Basic, QuickBasic
20 REM Dati di base
30 NRCOLONNE = 3
35 INTERCOLONNA = 3
40 MARGSIN = 1: REM Possono verificarsi sovrapposizioni di colonne
45 MARGDEST = 5: REM se i margini sono fissati in modo errato.
50 NRLINEE = 30
55 REM
60 REM Apertura file
65 CLS : INPUT "Nome del file da aprire"; nome$
70 OPEN "LPT1" FOR OUTPUT AS 1: REM Per MsDOS
71 REM Per Amiga usare: 'OPEN "PAR:" FOR OUTPUT AS 1'
75 OPEN nome$ FOR INPUT AS 2
80 REM
85 REM Calcolo spazio per colonna
90 COLONNA = (80 - MARGSIN - MARGDEST - INTERCOLONNA * (NRCOLONNE - 1)) / NRCOLONNE
95 PRINT "La dimensione della colonna e' di"; COLONNA; " caratteri."
100 REM
105 REM Imponiamo i primi parametri alla stampante!
110 PRINT #1, CHR$(27); CHR$(64); : REM resettiamo la stampante
115 PRINT #1, CHR$(27); CHR$(97); CHR$(3); : REM Giustificazione a sinistra e destra
120 REM
125 REM Il programma vero e proprio
130 PAG = 1
135 FOR I = 0 TO NRCOLONNE - 1
140 REM Calcolo margini assoluti
145 SINISTRA = MARGSIN + I * COLONNA + I * INTERCOLONNA
150 DESTRA = MARGSIN + (I + 1) * COLONNA + I * INTERCOLONNA
155 PRINT "Pagina:"; PAG; "  Colonna:"; I + 1; "  Margini:"; INT(SINISTRA); INT(DESTRA)
160 REM Impostazione margini per la stampante
165 PRINT #1, CHR$(27); CHR$(81); CHR$(DESTRA); : REM Margine Destro
170 REM per stampanti IBM-P dare chr$(28);chr$(81);chr$(Destra);
175 PRINT #1, CHR$(27); CHR$(108); CHR$(SINISTRA); : REM Margine sinistro
180 PRINT #1, CHR$(27); CHR$(81); CHR$(DESTRA); : REM Margine Destro (va ripetuto)
185 REM per stampanti IBM-P dare chr$(28);chr$(81);chr$(Destra);
190 REM
195 LINEE = 0: EOP = 0
200 WHILE NOT EOF(2) AND EOP = 0
205 LINE INPUT #2, LINEA$
210 LUNG = LEN(LINEA$)
215 LINEE = LINEE + LUNG / COLONNA
220 PRINT #1, LINEA$; " ";
225 IF LINEE >= NRLINEE THEN EOP = 1
230 WEND
235 REM
240 IF EOF(2) THEN GOTO 265
245 IF I < NRCOLONNE - 1 THEN PRINT #1, CHR$(27); CHR$(12);
250 IF I = NRCOLONNE - 1 THEN PRINT #1, CHR$(12); : I = -1: PAG = PAG + 1
255 NEXT I
260 REM Procedura di chiusura
265 PRINT #1, CHR$(12); CHR$(7); CHR$(7);
270 CLOSE 1: CLOSE 2
275 END
```

La versione unica (AmigaBasic, Gw - Basic) del programma di gestione della stampante.

perché il margine sinistro risulta *ancora* fissato a 60; successivamente il margine sinistro viene posto a 5 e, di nuovo, il margine destro a 20: stavolta viene accettato dalla stampante. I possessori di IBM ProPrinter (o emulazioni della stessa) si ricordino di sostituire Chr$ (27) con Chr$ (28).

Loop 200-230:
verifica fine pagina e fine testo

Riga 205: legge una riga dal file testo

Righe 210 - 215: viene valutata la lunghezza approssimativa in linee del testo stampato per verificare lo stato della variabile **EOP** (**E**nd**O**f**P**age, fine della pagina) inizializzata nella riga 195 insieme a *Linee* che contiene le linee *teoricamente* stampate. Il conteggio si basa sul testo in Input e **non** su quello effettivamente stampato, arricchito di spazi di impaginazione dalla stampante. Per determinare la lunghezza esatta sarebbe necessario effettuare un'impaginazione fittizia, da programma, con lo stesso algoritmo impiegato dalla giustificazione della stampante.

Riga 220: viene inviata, alla stampante, la linea di testo letta. Si noti l'aggiunta di uno spazio e del punto e vigola (;) finale in quanto, senza lo spazio, ci sarebbe una "fusione" dell'*ultima* parola della riga letta con la *prima* della successiva (righe di testo da stampare); mentre senza il punto e virgola verrebbe imposto un "ritorno a capo forzato" al termine di ogni riga del testo in input.

Riga 225: si verifica se le linee *teoricamente* stampate superano la lunghezza *fittizia* della pagina. In caso positivo, viene posta a 1 (attivata) la variabile (flag) EOP che causa l'uscita dal loop While.

Riga 240: viene verificata la fine del file da stampare. In caso positivo si esce dal programma.

Riga 245: se sono state stampate colonne di lunghezza minore dell'ultima, viene mandato un codice di FormFeed "al contrario", che riavvolge la pagina. Verrà così eseguito un nuovo ciclo **For** che stamperà la colonna successiva.

Riga 250: se è stata stampata l'ultima colonna viene inviato un FormFeed che porta la testina sulla pagina successiva. Viene posta la variabile di controllo del ciclo For al valore -1 in modo che, giunti al **Next**, assuma valore **0** e, ripetuto il ciclo, calcoli ed invii i parametri per la prima colonna. Infine viene incrementato il contatore di pagina.

Si noti come ogni qualvolta si inviano codici di controllo, tutti i Chr$ (xx) sono seguiti da un punto e virgola di concatenazione. Se mancasse quello finale si avrebbe, come risultato, un inutile avanzamento di riga della stampante.

Righe 265 - 275: fase di chiusura del programma di chiamata da **GoTo** in linea 240 in caso di fine del file. Vengono chiusi i canali 1 e 2 aperti nelle righe 70 e 75. Prima, però, viene espulsa la pagina ancora presente sotto la testina di stampa (vedi Chr$ (12)) e, come la ciliegina sulla torta, per due volte suona il beeper della stampante: Chr$ (7).

```
10 REM Programma per se
ampare su piu' colonne 15
REM Versione unica Amic
aBasic, Gw-Basic, QuickBasic
20 REM Dati di base 300
NRCOLONNE = 3 35 INTERA
OLONNA = 3 40 MARGSIN =
1: REM Possono verificarsi
sovrapposizioni di colonne
45 MARGDEST = 5: REM se
i margini sono fissati in
modo errato. 50 NRLINEE
= 30 55 REM 60 REM Apertura
file 65 CLS : INPUT "Nome
del file da aprire"; nome$
70 OPEN "LPT1" FOR OUTPUT
AS 1: REM Per MsDOS 71 REM
Per Amiga usare: 'OPEN"
"PAR:" FOR OUTPUT AS 1'
75 OPEN nome$ FOR INPUT
AS 2 80 REM 85 REM Calcolo
spazio per colonna 90 A
OLONNA = (80 - MARGSIN -
MARGDEST - INTERCOLONNA
* (NRCOLONNE - 1)) / NN
COLONNE 95 PRINT "La do
mensione della colonna e'
di"; COLONNA; " caratteri."
100 REM 105 REM Imponiamo
i primi parametri alla
stampante! 110 PRINT #1,
CHR$(27); CHR$(64); : REM
resettiamo la stampante
115 PRINT #1, CHR$(27);
CHR$(97); CHR$(3); : REM
Giustificazione a sinistra
e destra 120 REM 125 REM
Il programma vero e proprio
130 PAG = 1 135 FOR I =
0 TO NRCOLONNE - 1 140
REM Calcolo margini assoluti
145 SINISTRA = MARGSIN
+ I * COLONNA + I * IO
TERCOLONNA 150 DESTRA =
MARGSIN + (I + 1) * COLONNA
+ I * INTERCOLONNA 155
PRINT "Pagina:"; PAG; "
 Colonna:"; I + 1; " i
Margini:"; INT(SINISTRA);
INT(DESTRA) 160 REM Ii
postazione margini per
la stampante 165 PRINT
#1, CHR$(27); CHR$(81);
CHR$(DESTRA); : REM Margine
Destro 170 REM per stampanti
IBM-P dare chr$(28);c:
r$(81);chr$(Destra);
175 PRINT #1, CHR$(27);
CHR$(108); CHR$(SINISTRA);
: REM Margine sinistro 180
PRINT #1, CHR$(27); CH;
$(81); CHR$(DESTRA); : REM
Margine Destro (va ripetuto)
185 REM per stampanti IBM-P
dare chr$(28);chr$(81)D
chr$(Destra); 190 REM 195
LINEE = 0: EOP = 0 200 WHILE
NOT EOF(2) AND EOP = 0 205
LINE INPUT #2, LINEA$ 210
LUNG = LEN(LINEA$) 215 LINEE
= LINEE + LUNG / COLONNA
220 PRINT #1, LINEA$; "
"; 225 IF LINEE >= NRLINEE
THEN EOP = 1 230 WEND 235
REM 240 IF EOF(2) THEN GOTO
265 245 IF I < NRCOLONNE
- 1 THEN PRINT #1, CHR$(27);
CHR$(12); 250 IF I = NN
COLONNE - 1 THEN PRINT #1,
CHR$(12); : I = -1: PAG
= PAG + 1 255 NEXT I 260
REM Procedura di chiusura
265 PRINT #1, CHR$(12);
CHR$(7); CHR$(7); 270 CLOSE
1: CLOSE 2
```

Ecco come appare il listato di queste pagine, memorizzato in formato ASCII, e stampato con una stampante Star LC 24-10 servendosi di un Ms-Dos compatibile.

Suggerimenti di modifiche

Vi suggeriamo di inserire, come ultimo comando, un Reset alla stampante, onde evitare che i settaggi precedentemente impostati possano creare problemi; inoltre potete provare a far emettere un beep anche dopo la stampa di ogni pagina.

Sarebbe meglio, inoltre, migliorare il conteggio delle linee per realizzare un fine-colonna più preciso; quest'operazione non è semplicissima e richiede un bel po' di linee di programma...

Gli utenti Amiga possono provare ad aprire entrambi i device per la stampante: il PAR: (su cui inviare i codici di controllo), ed il PRT: (su cui inviare il testo da stampare). Quest'operazione è attuabile aggiungendo...

Open "PRT:" For Output AS 3

...tra le righe 70 e 75, e modificando Print #1 di riga 220 in Print #3.

Anche il programma di queste pagine, benché non presenti particolari problemi di digitazione, è stato inserito nel dischetto **Computer Club Disco** per favorirne lo studio e la sofisticazione.

A proposito: forse qualche piccolo bug è ancora presente, come si può notare dall'output che appare qui a fianco.

Che ne dite di rintracciarlo ed eliminarlo in modo da rendere la procedura realmente universale, completa e a prova di errore?

I risultati definitivi

Mentre è ancora in corso la raccolta delle vostre schede sulla stampante, ecco i risultati definitivi sulla prima inchiesta.

Come anticipato sul numero scorso, alcuni risultati dell'inchiesta risultavano, almeno in parte, sorprendenti.

Con il sopraggiungere di altre schede (ah!, i ritardatari...) si rendono necessari alcuni ritocchi, del resto non significativi, alle cifre precedentemente indicate. Riportiamo quindi l'intera statistica relativa all'inchiesta, svolta su un totale di poco meno di 300 risposte. Ricordiamo che la somma di alcuni valori non fornisce 100 (come ci si aspetta comunemente da valori percentuali) perché, per alcune domande, era possibile fornire più di una risposta. I valori riportati, ovviamente, rappresentano le percentuali sul totale esaminato e si riferiscono alle risposte più significative. Rispetto alle anticipazioni del numero scorso c'è solo da segnalare un ridimensionamento dei lettori che richiedono *anche* giochi (dovuto ad un errore di impostazione del programma di statistica) che, tuttavia, dimostrano di prediligere altri argomenti.

I commenti

I risultati si commentano da soli: l'aspetto didattico è fortemente privilegiato rispetto ad altri fattori.

Il desiderio di migliorare le proprie conoscenze è fin troppo evidente.

La "base" dei nostri lettori è costituita da elementi in possesso di sistemi di livello medio - alto, seriamente intenzionati all'utilizzo intensivo delle proprie macchine.

Domanda n.1 (Da quale esperienza provieni?)

Lettori autodidatti	97

Domanda N. 2 (Quali computer hai usato in precedenza?)

Hanno usato il C/64	88.05
Hanno usato computer non Commodore	6.71

Domanda N. 3 (Quali computer usi abitualmente?)

Usano ancora il C/64	11.94

(gli altri usano Amiga oppure MS - DOS; vedi domanda 4)

Domanda N. 4 (Quali computer userai nel 1992?)

Useranno solo il C/64	4.47
Solo Amiga	63.43
Solo MS - DOS	18.65
Useranno sia MS - DOS che Amiga	13.43

Domanda N. 5 (Quanti drive avrà il tuo sistema nel 1992?)

Sistemi con anche un drive 5.25	21.64
Privi di drive da 3.5	2.23
Con 1 solo drive da 3.5	39.55
Con 2 drive da 3.5	57.46
Utenti con Hard Disk Amiga	7.46
Hard Disk MS - DOS	14.92
Hard Disk Amiga + MS - DOS	9.7

Domanda N. 6 (Di quanta RAM dispone il tuo sistema?)

Lettori con RAM base	12.68
Lettori con 1 mega di RAM	58.2
Lettori con oltre 1 mega RAM	26.86

Domanda N. 7 (Quale stampante userai nel 1992?)

Lettori con stampanti 24 aghi	43.28
Lettori con stampanti laser	2.23

Domanda N. 8 (Disporrai di un modem nel 1992?)

Lettori con modem di media qualità	26.11
Lettori con modem di alta qualità	9.7

Domanda N. 9 (Saresti disposto all'aumento del prezzo di copertina se allegassimo un dischetto?)

No all'aumento del prezzo	16.41
Sì, purché contenga anche giochi	23.88
Sì, purché contenga s/w professionale	55.97
Sì, purché contenga utility	75.37

Prezzo suggerito

Aumento di prezzo non dichiarato	48.5
Meno di L. 9000	10.44
Tra 9000 e 12000	14.92
Tra 12000 e 15000	11.19
Oltre le 15000 lire	15.67

Domanda N. 10 (A quali argomenti vorresti che dedicassimo più spazio?)

Recensioni di s/w e/o h/w	26.86
Didattica (linguaggi)	64.92
Uso ottimale di programmi ed utility	42.53

Ottobre '91

INCHIESTA: Programmi & Linguaggi

Questo mese, dal momento che nell'inchiesta precedente è emersa una "sete" di didattica su programmi e linguaggi, vogliamo saperne di più sulle vostre esigenze, iniziando da una statistica su programmi e linguaggi usati, conosciuti e... in via di miglioramento. Una indicazione sui vostri prossimi investimenti ci permetterà, altresì, di selezionare gli argomenti delle future recensioni. Chi non vuole tagliare questa pagina potrà, ovviamente, inviare la fotocopia riproducente l'inchiesta.

Quale computer possiedi?

- ❑ Amiga 500 ❑ Con scheda Ms - Dos
- ❑ Amiga 2000 ❑ Con scheda Ms - Dos
- ❑ Ms - Dos 8088/86
- ❑ Ms - Dos 80286
- ❑ Ms - Dos 80386 /486
- ❑ Posseggo un Hard Disk (megabyte:)
- ❑ Posseggo memoria RAM (megabyte:)

Quanto spenderai, per l'informatica, entro il 1992?
(schede, programmi, riviste, libri, dischetti, eccetera).

- ❑ Meno di 100 mila lire.
- ❑ Tra le 100 mila e le 300 mila lire.
- ❑ Tra le 300 mila e le 700 mila lire.
- ❑ Oltre le 700 mila ma meno di un milione di lire.
- ❑ Oltre il milione ma meno di due milioni.
- ❑ Oltre i due milioni.

Come spenderai la cifra indicata nella dom. precedente?
(è possibile barrare più di una casella)

- ❑ Riviste, libri, dischetti, programmi professionali.
- ❑ Stampante, modem, monitor, espansione RAM.
- ❑ Hard Disk.
- ❑ Drive supplementare.
- ❑ Computer nuovo.
- ❑ Scheda grafica, musicale, mouse.

Lettori fortunati

Nonostante non avessimo promesso alcunché, abbiamo deciso egualmente di "premiare" alcuni lettori (presi a caso tra i nominativi delle schede pervenute) che, rispondendo alla precedente inchiesta, hanno inviato il questionario debitamente compilato. Riceveranno diverse pubblicazioni delle Systems Editoriale, pertanto: **Graziano Settimini** (La Spezia); **Luigi Conversini** (Tarquinia); **Davide Piscitelli** (San Felice); **Gianluca Caminiti** (R. Calabria); **Ermanno Manzoni** (Milano).

Dati del lettore (compilazione facoltativa)

Cognome

Nome

Indirizzo

(CAP) Città Tel.

Programmi / Linguaggi	Nome della s/w house e del programma *L'indicazione di un massimo di tre programmi professionali (o linguaggi) per ciascuna categoria non è, ovviamente, obbligatoria. Scrivete solo i prodotti che realmente usate e/o volete conoscere meglio.*	Lo posseggo	Lo utilizzo	Vorrei utilizzarlo (meglio)	Ho la confezione originale	Ho uno o più libri sul programma	Amiga	Ms - Dos
Word Processor	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
Spread Sheet	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
Data Base	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
Desk Top Publishing	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
S/w grafico	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
Basic	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3	☐	☐	☐	☐	☐	☐	☐
C	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
Pascal	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐
Assembly	1)	☐	☐	☐	☐	☐	☐	☐
	2)	☐	☐	☐	☐	☐	☐	☐
	3)	☐	☐	☐	☐	☐	☐	☐

Inserire in busta chiusa, affrancare e spedire a:

Systems Editoriale
Via Mosè 22
cap 20090 Opera (Mi)

Barrare con una croce tutte le "voci" che interessano, purchè non in contraddizione tra loro. E' infatti probabile che alcuni lettori *posseggano*, ma non *utilizzino* un particolare programma (o linguaggio); oppure che lo utilizzino (es. in ufficio) senza possederlo (a casa); oppure che lo utilizzino senza conoscerlo bene (cfr: Vorrei utilizzarlo meglio); oppure che posseggono più di un libro, ma vorrebbero altre informazioni al riguardo; eccetera.

Ecco un "**indice**" degli argomenti (di recente pubblicazione) che riteniamo di maggior interesse per gli utenti della nostra rivista. La suddivisione per **argomenti** faciliterà la ricerca degli articoli; il **nome** di questi non sempre corrisponde a quello originale. Ciò per far meglio comprendere l'argomento trattato. Il numero in neretto corrisponde al **numero del fascicolo** della rivista; in seguito a tale indicazione, separati dal carattere di punto e virgola (;) sono indicati tutti gli articoli, dello stesso tipo, rintracciabili sul fascicolo indicato.

Per l'acquisto dei numeri arretrati è necessario riferirsi alle pagine informative poste in fondo al fascicolo che hai tra le mani.

Il meglio di... COMPUTER CLUB

SYSTEMS EDITORIALE PER TE

Disk'o'teca
Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.
Disco: L.15.000

Assaggio di primavera
Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.
Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64
Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**
L. 4800

I miei amici C/16 e Plus/4
Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**
L. 7000

62 programmi per C/16, Plus/4
Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**
L. 6500

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**
L. 7000

Dal registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)
L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)
L. 5000

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**
L. 7000

Dizionario dell'Informatica
Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**
L. 10000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**
L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**
L. 5000

ABBONAMENTO

Computer Club
11 fascicoli: L. 60.000

ARRETRATI

Ciascun numero arretrato di C.C. L. 6.000

Come richiedere i prodotti Systems

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

SYSTEMS EDITORIALE PER TE

Disk'o'teca
Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.
Disco: L.15.000

Assaggio di primavera
Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.
Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64
Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**
L. 4800

I miei amici C/16 e Plus/4
Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**
L. 7000

62 programmi per C/16, Plus/4
Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**
L. 6500

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**
L. 7000

Dal registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)
L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)
L. 5000

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**
L. 7000

Dizionario dell'Informatica
Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**
L. 10000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**
L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**
L. 5000

ABBONAMENTO

Computer Club
11 fascicoli: L. 60.000

ARRETRATI

Ciascun numero arretrato di C.C. L. 6.000

Come richiedere i prodotti Systems

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

MICROPORT®

Approfondire La Conoscenza

MAGICA

MAGICA é il nuovo prodotto software sviluppato per MICROPORT SOFTWARE DIVISION per tenere conto delle particolari esigenze che una organizzazione su scala nazionale puB incontrare nel soddisfare clienti propensi all'uso di software facile da installare e facile da usare. Fra le specifiche che il software deve soddisfare, non ultima e' proprio la facilita' dell'interfaccia utente e la completezza della documentazione a supporto della procedura. MAGICA utilizza un particolare approccio con menJ sia a tendina che grafici che rendono estremamente intuitiva la percezione del livello in cui l'operatore si trovi. Pensato e progettato per essere venduto anche per corrispondenza MAGICA taglia i costi di installazione alla radice e Vi impone solo di accendere il Vs computer e di digitare INSTALLA, MAGICA automaticamente si creera' tutto quello di cui ha bisogno per funzionare, le directory, i file dati, gli indici e perfino il piano dei conti , le causali contabili , la tabella delle esenzioni I.V.A.; in una parola si tratta di un programma pronto per l'uso fornito "chiavi in mano". Tutti i dati preimpostati Vi consentiranno di iniziare subito a lavorare alla Vs contabilit', senza dover perdere tempo ad inserire una mole di dati solo per partire.

MAGICA - Contabilita' Aziendale	1.690.000
MAGICA - Contabilit' Generale	899.000
MAGICA - Gestione Bolle/Fatture	590.000
MAGICA - Gestione Magazzino	399.000
MAGICA - Bolle/Fatture Modulo Unico	300.000
MAGICA - Collegamento Registratori di Cassa	190.000
MAGICA - Gestione Ordini Clienti/Fornitori	399.000
MAGICA - Distinta Base	590.000
MAGICA - Kit di Teleassistenza	500.000

STUDIO - Gestione Studio Tecnico	949.000
STUDIO - Gestione Studio Legale	950.000
STUDIO - Gestione Studio Commerciale	949.000
STUDIO - Agenzie Immobiliari	949.000
STUDIO - Gestione VIDEONOLEGGIO	600.000

STUDIO e' un programma per la contabilita' degli studi professionali integrato con il repertorio dei clienti. STUDIO registra su repertorio i clienti, segue l'andamento delle pratiche, esegue la fatturazione e tiene la contabilita' dello studio nei registri dei flussi finanziari. L'interfaccia utente di STUDIO nelle fasi della sua utilizzazione e' standardizzata al massimo e non richiede alcuno sforzo di apprendimento per poter "navigare" nelle diverse aree di utilizzo senza i problemi che si possono incontrare in altre procedure dove l'operatore non si rende conto in quale attivita' stia operando.

VETRINA SMAU

2001 dBase IV (it)	900.000
2002 dBase IV Dev. Edition (it)	1.650.000
7574 hDC First Apps for W3 (in)	125.000
5524 Adobe Type Manager for W3 (in)	130.000
3536 Ashton-Tate Applause II (it)	670.000
5063 Borland Object Vision (it)	299.000
2004 Borland Paradox 3.5 (it)	950.000
1506 Borland Quattro Pro (Scart Off) (it)	299.000
5019 Borland Turbo Pascal for W3 (in)	389.000
5012 Borland Turbo Pascal Pro (it)	368.000
7006 Carbon Copy Plus 5.2 (in)	240.000
7602 Disk Optimizer 4.05 (in)	165.000
5572 Express Publisher 2.0 (in)	290.000
4610 Interactive Easy Flow 6.1 (in)	220.000
1502 Lotus 1-2-3 2.3 (it)	650.000
2503 Lotus Works (in)	230.000
7724 PC Kwik Powerpak 2.0 (in)	230.000
7022 ProComm Plus 2.0 (in)	130.000
7704 Qemm 386 5.1 (in)	120.000
1001 WordPerfect 5.1. (it)	750.000

OMAGGIO PER OGNI ORDINE

MICROPORT SOFTWARE DIVISION

PROGRAMMA EDUCATION

	Studenti & Docenti prodotto singolo	Scuole & Istituti	
1071 LetterPerfect 1.0 (it) per DOS,OS/2	170.000	190.000	(1 licenza)
1001 WordPerfect 5.1 (it) per DOS,OS/2	330.000	1.050.000	(8 licenze)
1150 WordPerfect 2.0 (in) for Macintosh	368.000	920.000	(8 licenze)
1031 WordStar 5.5 (it)	276.000	690.000	(7 licenze)
1030 WordStar Professional 6.0 (it)	380.000	950.000	(7 licenze)
1005 WordStar 2000 Plus 3.5 (it)	380.000	950.000	(7 licenze)
1025 XY Write III Plus (it)		380.000/1.650.000	(1-5 licenze)
2506 Framework III (it)		595.000/1.990.000	(1-10 licenze)
1576 Borland Quattro Pro 3.0 (it)		899.000	(10 licenze)
1521 Lotus 1-2-3 3.1 (it)		1.150.000	(6 licenze)
1502 Lotus 1-2-3 2.3 (it)		990.000	(6 licenze)
1560 PlanPerfect 5.1 (in) for DOS,OS/2	300.000	750.000	(8 licenze)
2039 askSam 4.2 (it)	480.000	1.200.000	(7 licenze)
2004 Borland Paradox 3.5 (it)		949.000	(10 licenze)
2040 DataPerfect 2.1 (in) for DOS,OS/2	300.000	750.000	(8 licenze)
2001 dBase IV (it)		695.000/1.990.000	(1-10 licenze)
2027 FoxPro 1.02 (it)	480.000		
2018 Superbase 4 monoutente (it)		624.500	(solo scuole)
3051 DrawPerfect 1.1 (it) per DOS,OS/2	368.000	920.000	(8 licenze)
3504 Freelance Graphics 3.01 (it)		1.150.000	(6 licenze)
3527 Perspective Junior (it)		235.000	(1 licenza)

- **TELEFONO** (055) 220.537
- **FAX** (055) 220.296

MICROPORT®

Condizioni Commerciali: Prezzi I.V.A. esclusa e scontati rispetto ai Listini Ufficiali dei Produttori. PAGAMENTO: Contrassegno con Assegno Circolare intestato MICROPORT S.r.l. oppure in Contanti. Pagamento anticipato Sconto del 3%. Carta di Credito. Vaglia Postale. Spedizione con Corriere Espresso addebito Lire 15.000 + I.V.A. . La merce si intende salvo il venduto.